Gestion et performance du calcul pour les expériences en physique et en astrophysique

Luisa Arrabito

Gestion et performance du calcul pour les expériences en physique et en astrophysique

Mémoire d'Habilitation à Diriger des Recherches
de l'Université de Montpellier

Spécialité Physique
École doctorale : Information, Structures, Systèmes

Dr. Luisa Arrabito

Laboratoire Univers et Particules de Montpellier
(UMR 5299)

Habilitation soutenue le 23 avril 2021 devant le jury composé de :

Pr. Denis Puy	Professeur Université de Montpellier, LUPM	Président
Dr. Johan Bregeon	Chargé de recherche, LPSC	Membre
Dr. Frédéric Piron	Directeur de recherche, LUPM	Membre
Dr. Dominique Boutigny	Directeur de recherche, LAPP	Rapporteur
Dr. Sabine Crépé-Renaudin	Directrice de recherche, LPSC	Rapporteur
Dr. Karl Kosack	Chargé de recherche, CEA-Saclay	Rapporteur

Table des matières

Introduction

Au cours des dernières décennies, les expériences en physique des hautes énergies (*High Energy Physics*, HEP) se sont dotées d'instruments de plus en plus performants produisant une très grande quantité de données. Dans ce domaine, le *Large Hadron Collider* (LHC) est l'instrument qui génère actuellement le plus grand volume de données, i.e. 100 Pétaoctets par an. Le stockage, le traitement et la distribution de ces données représentent un défi majeur pour ces expériences en termes d'infrastructure de calcul, modèle de calcul et solutions logiciels. La solution choisie par le LHC pour faire face à ce défi a été celle d'un environnement distribué à grande échelle de type grille de calcul.

Malgré le succès de la grille pour le traitement des données du LHC, le modèle actuel ne permet pas de supporter l'augmentation des données prévue en 2026 avec la période à haute luminosité (*High Luminosity LHC*), estimées à plusieurs Exaoctets par an. Dans ce contexte, la communauté HEP a entamé un vaste programme de R&D dans plusieurs domaines, comme les infrastructures de calcul, l'optimisation des programmes, la simulation, l'apprentissage automatique, etc.

La nouvelle génération d'instruments dans les domaines de l'astrophysique et de l'astroparticules (CTA, LSST, SKA, Euclid, LISA), dont la mise en service est prévue entre 2020 et 2030, suit la même tendance en termes de quantité de données générées et doit faire face à des défis similaires. C'est dans ce contexte et en particulier dans le cadre du projet CTA (*Cherenkov Telescope Array*), que je me suis intéressée aux problématiques d'optimisation de l'utilisation des ressources de calcul. Le projet CTA a pour objectif la construction et l'exploitation du plus grand réseau de télescopes Cherenkov dédié à l'astronomie gamma de très haute énergie. Il sera constitué d'environ une centaine de télescopes Cherenkov sur les sites de La Palma (Canaries, Espagne) et Paranal (ESO, Chili).

Dans ce manuscrit d'habilitation, j'ai choisi de traiter deux aspects complémentaires relatifs à l'optimisation du calcul à travers mes contributions dans le cadre de CTA. Le premier aspect concerne l'optimisation du point de vue de la soumission et de l'ordonnancement des tâches sur une infrastructure distribuée (*Workload Management System* ou WMS). J'aborderai cette problématique à travers le projet `CTA-DIRAC` en présentant les caractéristiques fondamentales du WMS du framework DIRAC et son application aux traitements de CTA.

D'un tout autre point de vue, un deuxième levier pour améliorer les performances du calcul consiste à optimiser les codes qui sont exécutés dans ces environnements distribués. J'ai eu l'occasion d'approfondir cette thématique à travers ma contribution au projet `CTAOptSim`, effectuée dans le cadre d'une collaboration entre astrophysiciens et informaticiens et dont l'objectif est la réduction du temps de calcul du code de simulation de cascades atmosphériques `CORSIKA`. L'idée de base consiste à tirer le meilleur profit des capacités matérielles

des processeurs modernes tout en garantissant la portabilité du code. À travers ce projet, je mettrai en exergue l'intérêt des collaborations entre physiciens et informaticiens pour les deux communautés.

Pour finir, je conclurai ce manuscrit en décrivant les prospectives et les enjeux du calcul, i.e. traitement massif de données et simulation, dans le domaine de la physique des hautes énergies et de l'astrophysique ainsi que les synergies potentielles avec la recherche en informatique.

Chapitre 1

CTA, le premier Observatoire de rayons gamma de très haute énergie

1.1 Présentation

Le projet CTA (*Cherenkov Telescope Array*) se propose de concevoir, construire et exploiter le plus grand réseau de télescopes Cherenkov dédié à l'astronomie gamma de très haute énergie. CTA permettra d'obtenir des images du ciel avec une sensibilité (flux minimum détectable en provenance d'une source ponctuelle stable) et une résolution angulaire dix fois supérieures aux instruments actuellement en activité, tels H.E.S.S., MAGIC ou Veritas. Les objectifs scientifiques de CTA sont très ambitieux tant pour la compréhension des phénomènes astrophysiques de haute énergie que du point de vue de la physique fondamentale. Ceux-ci sont articulés autour d'une dizaine de projets scientifiques clés (*Key Science Project*, KSP), détaillés dans le livre *Science with the Cherenkov Telescope Array* [1]. La réalisation de ce vaste programme scientifique nécessite un nombre conséquent d'heures d'observation, d'une part pour monitorer des objets identifiés sur des longues périodes et d'autre part pour réaliser des relevés détaillés.

CTA utilise la technique des télescopes imageurs Cherenkov atmosphériques pour une détection indirecte des rayons gamma d'origine astrophysique. Cette technique, utilisée également par les instruments actuels, consiste à détecter les photons émis par effet Cherenkov lors des cascades (ou gerbes) de particules produites par l'interaction des rayons gamma avec l'atmosphère terrestre.

Deux caractéristiques clés permettent à CTA d'atteindre des performances inégalées. Tout d'abord CTA sera constitué d'environ 10 fois plus de télescopes par rapport aux instruments actuels, pour un nombre total de 118, ce qui augmente considérablement la surface effective et permet d'atteindre une sensibilité d'au moins un ordre de grandeur supérieure. Deuxièmement, l'utilisation de télescopes de 3 tailles différentes (LST, *Large Size Telescope* ; MST, *Medium Size Telescope* ; SST, *Small Size Telescope*) permettra de couvrir un large domaine en énergie, de 30 GeV à 300 TeV. Afin d'assurer une couverture totale du ciel, CTA disposera de deux réseaux de télescopes implantés dans les deux hémisphères. Dans l'hémisphère Nord le site Roque de los Muchachos à La Palma (Iles Canaries) comportera 19 télescopes (4 LST, 15 MST), dont le premier LST est déjà en fonctionnement. Les observations depuis le site Nord permettront en particulier l'étude des objets extragalactiques. Dans l'hémisphère Sud,

c'est le site ESO [1] du Paranal au Chili qui a été retenu. Il disposera de 99 télescopes (4 LST, 25 MST, 70 SST) et permettra d'observer les régions intérieures du plan galactique.

Enfin, CTA aura la particularité de fonctionner sur un mode d'observatoire. Cela signifie qu'une partie du temps d'observation sera ouvert à proposition et l'ensemble des données sera publique après une période de temps propriétaire prédéfinie (probablement 1 an).

Pour ma part, j'ai rejoint CTA dès mon arrivée au LUPM en 2011. Mes activités s'inscrivent dans le cadre du projet *Computing* et plus précisément du *Data Processing and Preservation System* (cf. Figure 1.3), que je vais présenter dans la Section 1.4. En 2011, les activités autour du calcul étaient menées dans le cadre du *Work Package Data Management*. Dans ce cadre, j'ai participé activement aux réunions face-à-face et aux discussions qui ont conduit à l'élaboration du modèle de calcul de CTA, à savoir l'estimation des besoins de calcul et de stockage, les stratégies de réplication et de traitement de données ainsi que l'organisation de la distribution des données et des traitements sur un ou plusieurs centres de calcul. Dans la Section 1.4.3, je présenterai le modèle retenu, basé sur une infrastructure distribuée sur 4 centres de calcul. En lien avec cette activité, j'ai très tôt proposé d'évaluer le framework DIRAC [2] [3], présenté dans le Chapitre 2, pour la gestion des traitements de données de CTA [4], en lançant le projet `CTA-DIRAC`.

Ce projet, exposé en détail dans le Chapitre 3, a consisté d'abord à développer un démonstrateur et ensuite un prototype de système de gestion de tâches, basé sur DIRAC. À travers ce projet, le LUPM a pris la responsabilité du composant en charge de la gestion de tâches de calcul, nommé `CWMS` (*Computing Workload Management System*) dans l'architecture du *Data Processing and Preservation System* de CTA, cf. Figure 1.4. Je mène ce projet, en étroite collaboration avec J. Bregeon (Chargé de Recherche au LUPM entre 2013 et 2019 et actuellement au LPSC) et les membres du Consortium DIRAC, en particulier R. Graciani Diaz (Université de Barcelone) au début du projet et A. Tsaregorodtsev (CPPM) par la suite.

En parallèle à cette activité d'évaluation et de développement, j'ai également assuré, en binôme avec J. Bregeon, la production des simulations pendant toute la phase de préparation de CTA (2011-2017) et la phase actuelle de pré-construction (2018-2020). Cette action a été menée en exploitant le prototype `CTA-DIRAC` et les ressources de la grille EGI. Dans le Chapitre 3, j'expliquerai en particulier le rôle de `CTA-DIRAC` dans la réalisation de ces simulations et le rôle essentiel de ces dernières dans l'optimisation du réseau.

Depuis 2017, j'ai également démarré une collaboration avec l'équipe DALI du LIRMM [2], autour de l'optimisation du code de simulation de cascades atmosphériques `CORSIKA`, utilisée par CTA et de nombreuses autres expériences. Ce projet, `CTAOptSim`, présenté en détail dans le Chapitre 4, a été financé pendant deux ans par le PEPS Astro-Informatique (voir Annexe A.2) et fait actuellement l'objet d'une thèse que je co-encadre avec l'équipe DALI du LIRMM.

1.2 Le principe de détection

Les rayons gamma de haute énergie interagissent avec les atomes de l'atmosphère et produisent des cascades de particules. Lorsque ces particules dépassent la vitesse de la lumière dans l'air, des photons sont émis par effet Cherenkov en formant un cône de lumière bleu. La

1. European Organisation for Astronomical Research in the Southern Hemisphere
2. `http://www.lirmm.fr/`

lumière Cherenkov ainsi produite se propage à travers l'atmosphère jusqu'au sol où elle est détectée par des télescopes à imagerie Cherenkov. Ces télescopes sont équipés d'une caméra placée dans le plan focal, qui enregistre les flash de lumière Cherenkov à l'aide de tubes photomultiplicateurs arrangés en pixels et d'une électronique de lecture rapide (le signal Cherenkov ayant une durée d'environ 10 ns). Actuellement, 6 prototypes de caméras existent : 1 pour les LST, 2 pour les MST et 3 pour les SST. La caméra proposée par la collaboration française pour les MST (NectarCAM) dispose ainsi de 1855 pixels arrangés en 265 modules.

CTA utilise cette technique pour la détection indirecte des rayons gamma d'origine astrophysique, tel qu'illustré sur la Figure 1.1. Cependant, la plupart des événements détectés est due au bruit de fond des rayons cosmiques, avec un flux d'environ un facteur 10^3 supérieur au flux des rayons gamma. Différentes techniques, exploitant certaines caractéristiques des images des gerbes dans les télescopes, sont utilisées pour la rejet de ce bruit de fond.

Le nombre de photons Cherenkov produits par unité de longueur de parcours et par intervalle de longueur d'onde pour une particule de charge ze est donné par :

$$\frac{d^2N}{dxd\lambda} = \frac{2\pi\alpha z^2}{\lambda^2}\left(1 - \frac{1}{\beta^2 n^2(\lambda)}\right) \tag{1.1}$$

$$cos(\theta_c) = \frac{1}{n\beta} \tag{1.2}$$

où N est le nombre de photons de longueur d'onde λ, α la constante de structure fine, n l'indice de réfraction du milieu et βc la vitesse de la particule incidente. Quant à l'angle d'ouverture du cône de lumière Cherenkov, il est donné par la formule 1.2. On note en particulier que l'intensité du rayonnement Cherenkov et l'angle d'ouverture du cône dépendent de la vitesse de la particule incidente ainsi que de l'indice de réfraction du milieu.

La majeure partie des photons Cherenkov étant produite à des altitudes supérieures à 10 km, il est important de connaître la composition et la densité de l'atmosphère au-dessus de cette altitude pour comprendre la génération de la cascade. Ensuite, lorsque les photons Cherenkov se propagent à travers les couches plus basses de l'atmosphère, une partie d'entre eux est absorbée. De plus, leurs trajectoires sont également déviées par effet de la réfraction. Une description précise de l'atmosphère à différentes altitudes est donc primordiale à la fois pour comprendre la génération de la cascade et son image Cherenkov au sol.

Les photons Cherenkov qui arrivent au sol sont ensuite détectés par un ou plusieurs télescopes. Afin de couvrir une très large gamme d'énergie, allant de 30 GeV à 300 TeV, CTA emploie des télescopes de 3 tailles différentes. Aux basses énergies, la densité et l'empreinte au sol des photons Cherenkov sont assez faibles. Pour des cascades générées par des rayons gamma de 30 GeV, on compte seulement quelques centaines de photons Cherenkov dans un rayon de 100 m autour du point d'impact et moins de 1 ph/m^2 à 200 m de celui-ci. Grâce à leur grande surface collectrice, les télescopes de grande taille permettent de détecter efficacement ce type d'événements.

En revanche, aux plus hautes énergies la densité des photons Cherenkov et l'empreinte au sol sont plus importantes. A 30 TeV, la densité de photons au centre de la gerbe atteint quasiment 10^5 ph/m^2, et même à 500 m du centre celle-ci, elle est encore supérieure à 100 ph/m^2. L'empreinte au sol est également très grande et couvre plusieurs km^2. Pour la détection de ces événements, les télescopes de petite taille sont les plus adaptés, car ils peuvent être

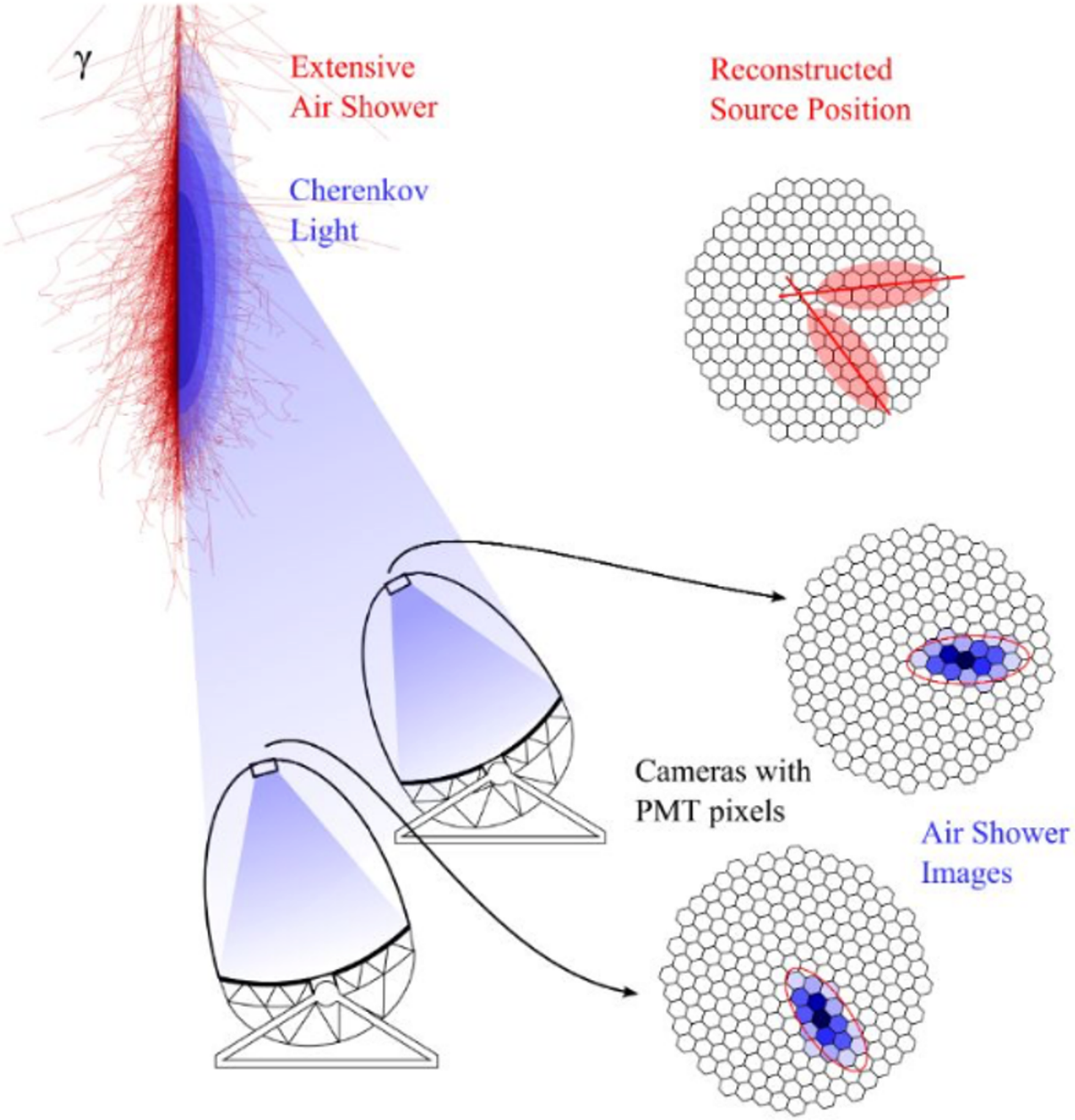

FIGURE 1.1 – Principe de détection des rayons gamma par la technique des télescopes imageurs Cherenkov utilisée dans CTA [5]. Un rayon gamma de haute énergie traversant l'atmosphère, génère une cascade de particules. Lorsque ces particules dépassent la vitesse de la lumière dans l'air, des photons sont émis par effet Cherenkov. Les télescopes imageurs collectent la lumière Cherenkov à l'aide de miroirs qui focalisent cette lumière sur une caméra, permettant ainsi d'obtenir une image de la cascade atmosphérique. En bas à droite, les images de la cascade sur les caméras et en haut à droite, la superposition de ces images permettant la reconstruction de la position du rayon gamma initial.

déployés en grand nombre et couvrir une grande surface tout en minimisant les coûts. Sans surprise, les télescopes de taille moyenne permettent de détecter efficacement les événements aux énergies intermédiaires.

Pour des gerbes générées par des rayons gamma, les images formées par les photons Cherenkov dans les caméras ont en bonne approximation une forme d'ellipse. La reconstruction stéréoscopique des paramètres de ces images permet de déduire les caractéristiques de la particule incidente, comme la direction d'arrivée, l'énergie, l'altitude de la première interaction et la nature de la particule. En particulier, les images des gerbes de type gamma ont une forme elliptique caractéristique, alors que celles des gerbes hadroniques, induites par le bruit de fond des rayons cosmiques, ont une forme pas très bien déterminée, tel qu'on peut le voir sur les images de la Figure 1.2. Des algorithmes d'apprentissage automatique ou profond sont généralement utilisés dans le pipeline de traitement pour la classification des événements de type gamma ou hadron.

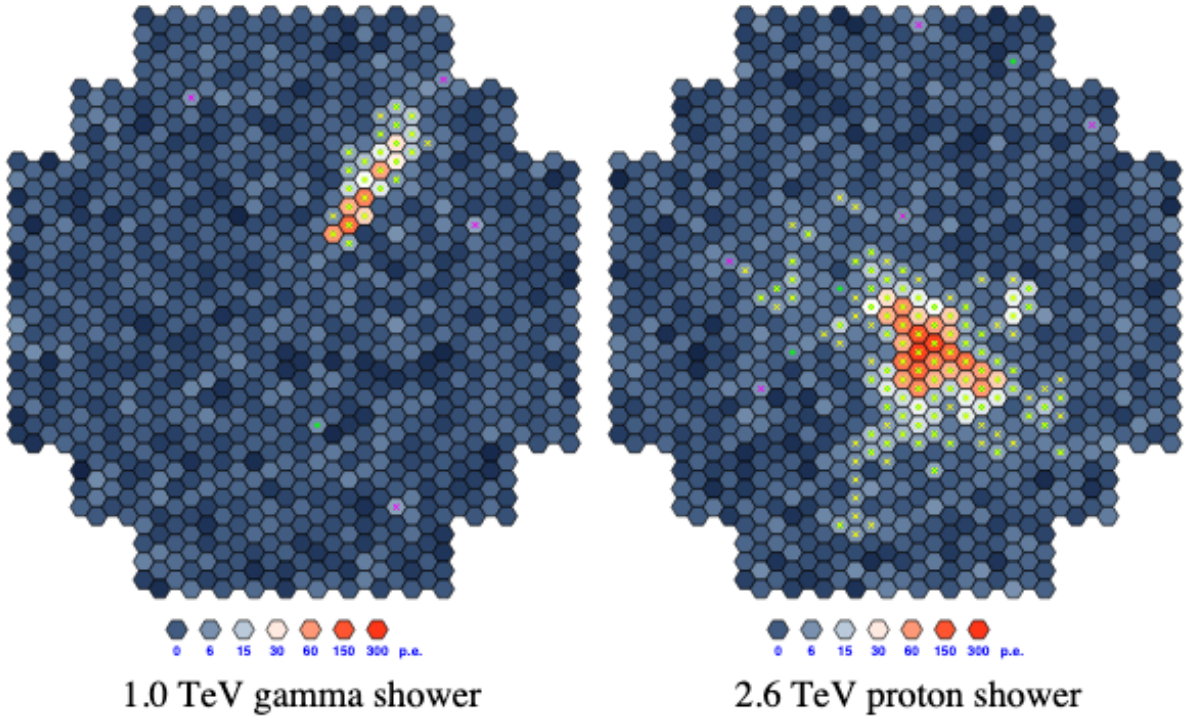

FIGURE 1.2 – Différence entre l'image caméra d'une gerbe induite par un photon gamma de 1 TeV (à gauche) et l'image d'une gerbe induite par un proton de 2.6 TeV (à droite) [6].

1.3 Organisation

La gouvernance de CTA diffère par rapport aux expériences en physique des hautes énergies. Deux entités distinctes participent à la construction et aux opérations de CTA avec des rôles différents : l'observatoire (CTAO, *CTA Observatory*) et le consortium (CTAC,

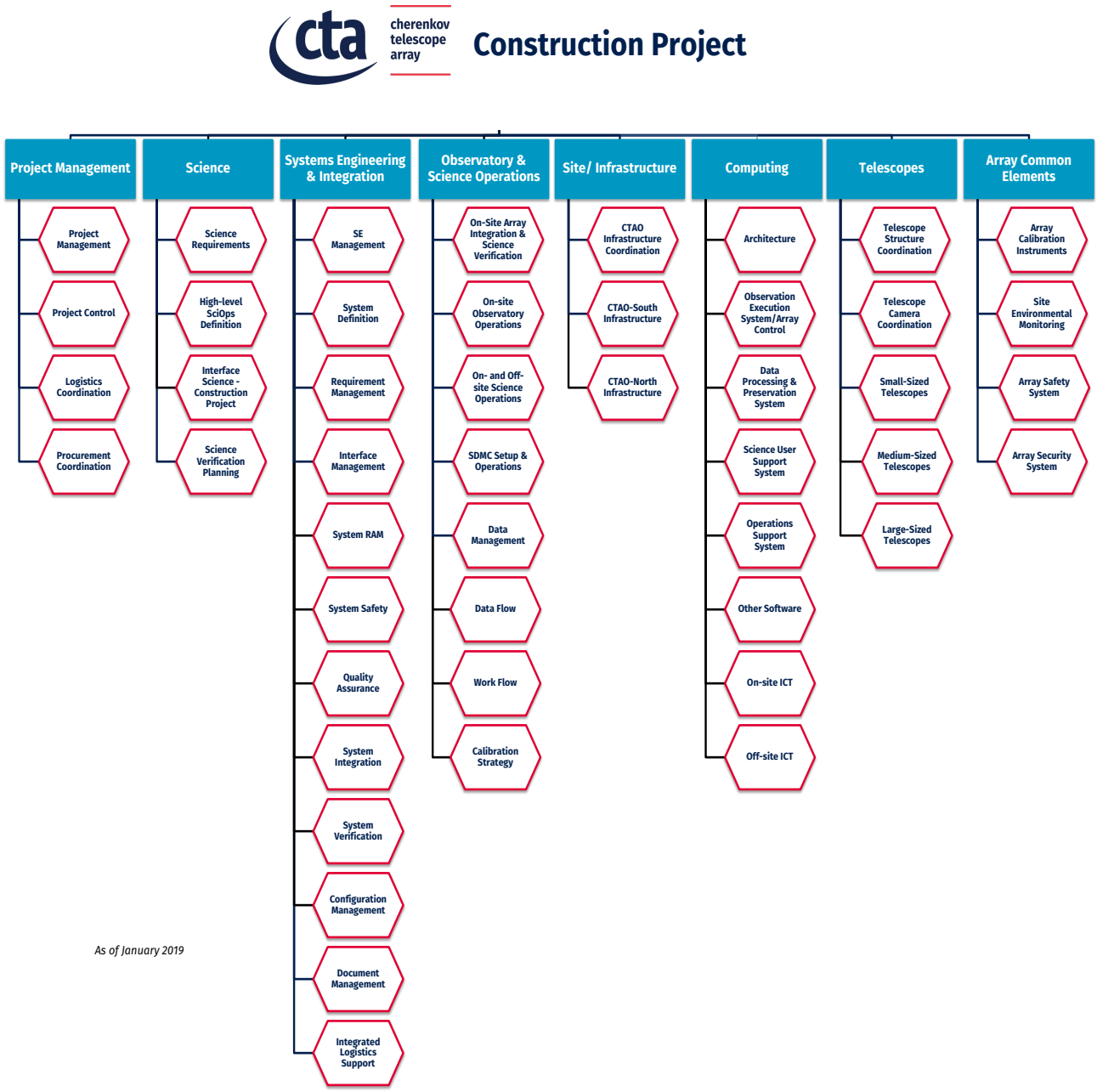

FIGURE 1.3 – Structure du projet de construction de CTA avec sa décomposition en 8 projets principaux [7].

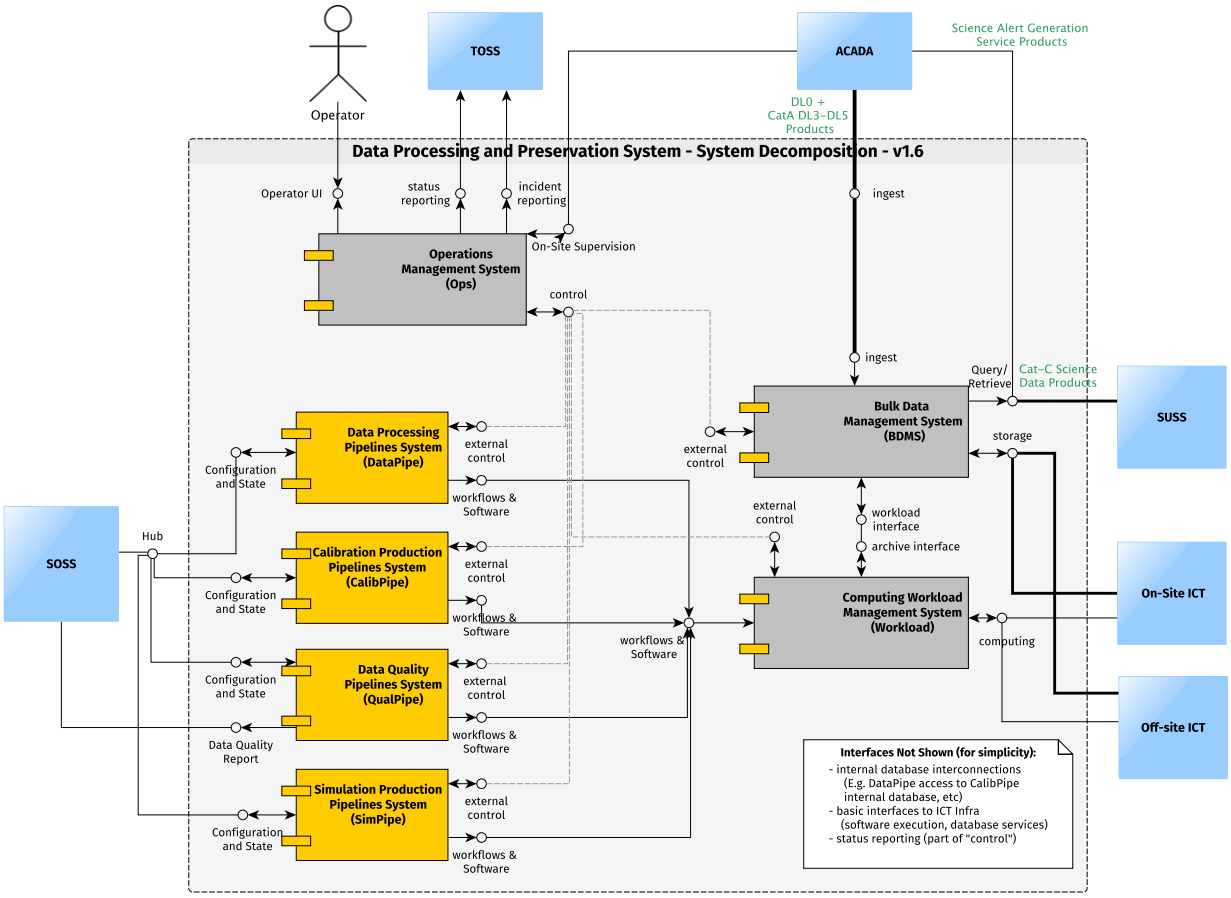

FIGURE 1.4 – Architecture du *Data Processing and Preservation System*. En jaune, les pipelines de traitement, calibration, contrôle de qualité de données et de simulation. En gris, les composants pour l'orchestration des tâches de traitement, la gestion des données et les opérations. Les flèches représentent les interfaces entre les différents composants mais également celles avec les systèmes externes, tels que ACADA (*Array Control and Data Acquisition System*) en charge du contrôle des télescopes, de l'exécution des observations et de l'acquisition des données, SUSS (*Science User Support System*) en charge des workflows d'analyse scientifique de haut niveau et point d'accès pour les utilisateurs, TOSS (*Technical Operations Support Systems*), SOSS (*Science Operations Support System*) et les centres de calcul sur site et hors ligne (*On-Site, Off-site Information and Communications Technology*).

CTA Consortium). L'observatoire est l'entité légale (ERIC) responsable de la conception, la construction et l'opération de CTA et travaille en étroite collaboration avec le consortium, qui rassemble près de 1500 membres appartenant à environ 200 instituts dans 31 pays. Les scientifiques et ingénieurs du consortium travaillent depuis environ 15 ans à la conception de CTA et au développement des prototypes pour les différents composants instrumentaux et logiciels. En outre, le consortium a élaboré le programme scientifique de CTA sous la forme de projets scientifiques clés. Au final, les contributions des instituts membres du consortium à la construction et aux opérations de l'observatoire se traduisent sous la forme de contributions en nature (*Inkind Contribution*). Les financements sont internationaux pour un coût total avoisinant les 500 millions d'euros, dont la France est l'un des contributeurs majeurs, en apportant environ 15% du budget global. Après une phase de préparation qui s'est étalée entre 2010 et 2017, le projet est actuellement en phase de pré-construction, avec le le premier télescope LST installé sur le site Nord et actuellement en cours de mise en service. Le début des opérations est prévu en 2023.

Le projet de construction piloté par CTAO est structuré avec une répartition du travail sur 8 projets, tel que détaillé en la Figure 1.3. Dans cette répartition, le projet *Computing* et en particulier le *Data Processing and Preservation System*, a en charge tous les aspects liés à l'archivage, le traitement et la distribution de données de CTA.

1.4 Gestion et traitement des données

La décomposition fonctionnelle du *Data Processing and Preservation System* (DPPS) en sous-systèmes est illustrée sur la Figure 1.4. Les interfaces entre les différents composants du DPPS, ainsi qu'avec les composants externes sont également représentées. Dans cette section, je vais présenter le modèle de calcul de CTA ainsi que le rôle des principaux composants du DPPS, tels que les pipelines, le système d'archivage et le gestionnaire de tâches de calcul (`CWMS`).

1.4.1 Les pipelines

Le traitement de données de CTA est réalisé par une série d'étapes qui transforment les données brutes des caméras en produits scientifiques de haut niveau. Ces étapes sont réalisées par différents pipelines : le pipeline de simulations Monte-Carlo, le pipeline principal de traitement, puis les pipelines d'analyse scientifique.

Aux différentes étapes du traitement correspondent 5 niveaux de données (*Data Level*, DL0-DL5) :

- DL0 : les données brutes des caméras
- DL1 : les données des caméras calibrées
- DL2 : les paramètres reconstruits des cascades, i.e. l'énergie, la direction et l'identifiant de la particule incidente
- DL3 : la liste des candidats gamma
- DL4 : les spectres en énergie ou les cartes du ciel

- DL5 : les données *legacy* de l'observatoire, telles que les cartes du ciel ou le catalogue des sources

Les données de bas niveau (DL0-DL3) sont produites par l'observatoire à travers le pipeline principal de traitement et le pipeline de simulation. Les données de haut niveau (DL4-DL5) sont produites par les pipelines d'analyses scientifiques et distribuées à la communauté à travers les services de l'Observatoire Virtuel [3].

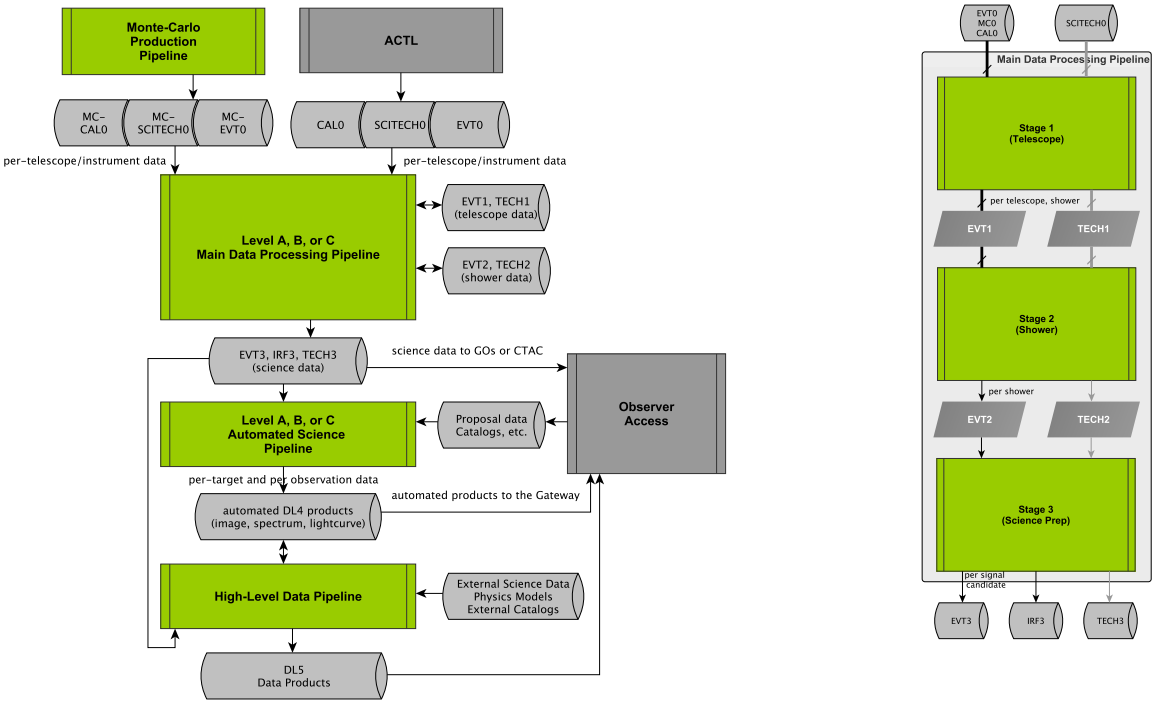

FIGURE 1.5 – À gauche, la vue globale des différents pipelines (Monte-Carlo, traitement et analyse). À droite, une vue plus détaillée du pipeline principal de traitement composé de 3 étapes : 1) étalonnage, nettoyage et paramétrisation des images par télescope ; 2) reconstruction stéréoscopique des paramètres des cascades à partir des images de plusieurs télescopes ; 3) discrimination des événements de type gamma ou hadron.

Les trois étapes principales du pipeline principal de traitement sont représentées à droite de la Figure 1.5. Les images de chaque caméra sont d'abord étalonnées, nettoyées et paramétrisées. Ensuite, plusieurs images sont considérées ensemble pour la reconstruction des paramètres de la cascade, tels la direction, l'énergie et les paramètres permettant de discriminer les événements de type gamma ou hadron. La dernière étape applique une sélection sur les événements et produit une liste de candidats gamma (DL3). Ces produits constituent les principales données d'entrée pour les analyses scientifiques de haut niveau et sont les plus fréquemment requis par les utilisateurs.

Ces pipelines existent en 3 variantes (niveaux A, B, C) ayant des contraintes différentes en termes de sensibilité, délais pour la production des résultats et environnement d'exécution. Les fonctionnalités et l'architecture sont les mêmes pour les différentes variantes, mais

3. `https://ivoa.net/`

l'implémentation et le framework sont très différents. En particulier, les pipelines de niveau A sont exécutés sur site (*on-site*) et en temps réel. Les contraintes sur la sensibilité sont réduites, mais elles doivent produire des résultats avec un temps de latence minimal, notamment pour la génération d'alertes. En particulier, ils doivent permettre d'identifier une source potentielle dans le champ de vue en moins de 30 s. Les pipelines de niveau B sont également exécutés sur site mais en mode *batch*. Ils doivent fournir des résultats de qualité, avec une sensibilité inférieure d'un facteur 2 par rapport aux spécifications, au plus tard 10 heures après la fin de l'observation. Enfin, le pipelines de niveau C sont exécutés dans les centres de calcul (*off-site*) et ils doivent fournir des données utilisables pour publication, en respectant donc les engagements de l'observatoire en terme d'incertitudes systématiques, dans un délai maximal de 2 mois.

Par ailleurs, le pipeline de simulation, en partant d'une description précise de l'instrument et des conditions d'observation, produit des données simulées de niveau DL0 qui reflètent le mieux possible les données réelles acquises par les télescopes. Ces données sont ensuite injectées dans le pipeline de traitement principal pour générer les fonctions de réponse de l'instrument (*Instrument Response Function*, IRF). Les fonctions de réponse caractérisent les performances de CTA, et permettent d'estimer les propriétés physiques de la particule incidente (nature, direction, énergie, temps d'arrivée) à partir des quantités reconstruites de la cascade atmosphérique. Elles sont exprimées en termes de surface effective, résolution angulaire et résolution en énergie. Les fonctions de réponse sont ensuite injectées, avec la liste de candidats gamma, dans le pipeline d'analyse scientifique. En comparant les données observées avec des modèles spectro-morphologiques, le pipeline d'analyse scientifique permet d'estimer le flux, l'indice spectral, la position et la morphologie des sources gamma et de produire les données de plus haut niveau, tels les spectres en énergie, les cartes du ciel ou le catalogue des sources.

1.4.2 Besoins en stockage et calcul

Les différentes étapes de traitement opèrent une réduction importante du volume de données, cf. Figure 1.6. Par conséquent, le volume de données global est déterminé par les données de niveau DL0 qui sont prépondérantes. Ce volume est calculé à partir du débit de données des caméras des sites Nord et Sud, estimé respectivement à 0.318 Go/s et 0.538 Go/s, après une réduction appliquée sur site. Le temps d'observation annuel, correspondant à des conditions d'observation du ciel nocturne, est d'environ d'environ 1300 heures, ce qui correspond à approximativement 4 Po de données brutes par an. En cumulant les différentes versions et copies de données prévues par le modèle de calcul (cf. Section 1.4.3), le volume total de données est estimé à environ 27 Po par an. Les simulations Monte-Carlo génèrent également une quantité très importante de données. Il est prévu qu'environ 20 Po de données de simulation soient archivées de manière permanente chaque année. Les besoins en termes de stockage augmentent tout au long de la vie de l'observatoire selon le profil de la Figure 1.7.

Les besoins CPU pour le traitement de données et la simulation sont également très importants. Il est prévu que les données d'une nuit d'observation soient traitées en moins d'un jour, ce qui devrait nécessiter environ 2000 coeurs tout au long de l'année. En plus de ce traitement, il est prévu au moins un re-traitement par an de toutes les données archivées, ce

Data occupying storage - Planned Model (Data)					
	Event Size	Data Access	Disk replicas of each version	Number of versions	Number of Tape Copy
Raw (DL0)	14000 to 41000 bytes depending on camera	Write once, low read rate	10% kept on disk (Tape cache is 10% of the global data volume)	1	1+1 (backup)
Calibrated (DL1)	20% to 100% of RAW prior to any data volume reduction		0	0	0
Reconstructed (DL2)	10% of RAW camera data prior to any data volume reduction	New version per year. Low read rate.	100% kept on disk for the last version, 0% for the old version	2	1+1 (backup for last version only)
Reduced (DL3)	1% of DL2	High read rate	1 (100%)	2	1+1 (backup)
Science (DL4)	0.1% of DL2	High read rate	1 (100%)	2	1+1 (backup)
Observatory (DL5)	0.1% of DL2	High read rate	1 (100%)	2	1+1 (backup)
MC Data	100% of Observation data (Min 5PB, Max 20PB)	Read/Write	100% during commissioning phase (3 years), 1PB afterwards	1	1+1 (backup)

FIGURE 1.6 – Les niveaux de données de CTA avec leur volume, modalité d'accès, nombre de versions et de copies prévues par le modèle de calcul.

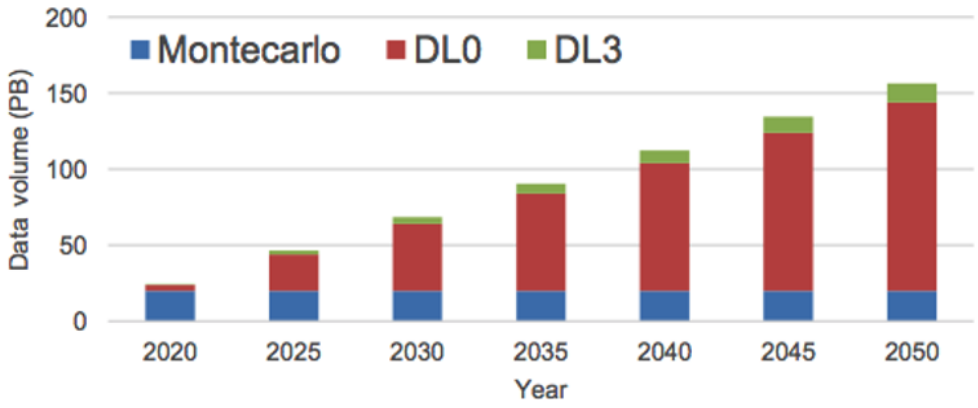

FIGURE 1.7 – Estimation du volume global des données de niveaux DL0-DL3 et Monte-Carlo au cours des 30 ans d'exploitation de CTA.

qui va demander en moyenne 6000 coeurs supplémentaires pendant la période de traitement. Pendant la phase de construction, il est prévu que le re-traitement n'excède pas la durée d'un mois, alors qu'après cette phase, la durée du re-traitement sera prolongée d'un mois chaque année. Enfin, les simulations Monte-Carlo pour la production des fonctions de réponse nécessitent des ressources supplémentaires, de l'ordre quelques milliers de coeurs tout au long des observations.

1.4.3 Modèle de calcul

Afin de concevoir le modèle de calcul pour l'archivage et le traitement des données de CTA, une étude comparative a été menée entre une solution centralisée dans laquelle les ressources de calcul et de stockage sont hébergées dans un seul centre et des solutions distribuées dans lesquelles les ressources sont réparties entre plusieurs centres. Le modèle actuellement retenu, qui représente le meilleur compromis entre qualité de service, sécurité des données et coût, est un modèle distribué qui comprend 4 centres de calcul (cf. Figure 1.8). Le modèle prévoit que les 4 centres se répartissent de manière équitable l'archivage et le traitement des données. Chaque niveau de données a une volumétrie et un profil d'accès différent. Les données de niveau DL0 sont les plus volumineuses, elles sont écrites une seule fois et le taux d'accès en lecture est relativement faible. En revanche, les données de haut niveau (DL3-DL5) sont peu volumineuses avec un taux d'accès en lecture élevé.

Plusieurs versions de données seront également maintenues. En particulier, il est prévu de re-traiter au moins une fois par an toutes les données archivées avec la version des algorithmes la plus à jour et la plus performante. Les deux dernières versions des données de niveaux DL2-DL5 seront sauvegardées.

Afin d'assurer la sécurité des données et un accès efficace, le modèle prévoit plusieurs copies des données. Le nombre de copies prévu pour chaque niveau est détaillé sur la Figure 1.6. En fonction de la volumétrie et du profil d'accès, les différentes copies seront sauvegardées sur disque ou sur bande magnétique. Pour des raisons de coût, le stockage sur bande est privilégié lorsque l'accès à ces données n'est pas intensif. Au total, environ 80% des données devrait être sauvegardé sur bande. Toutes ces versions et copies seront distribuées de manière

équitable entre les 4 centres de calcul.

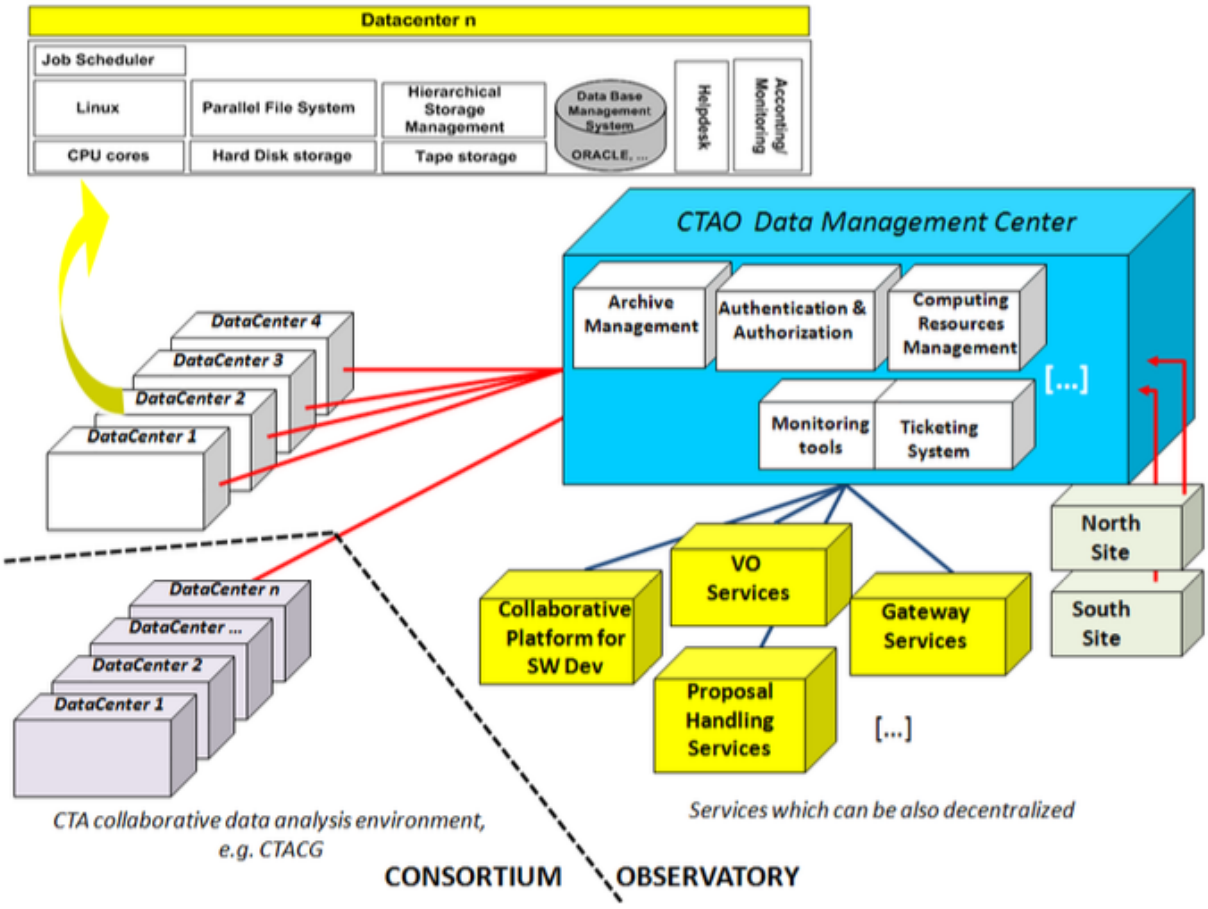

FIGURE 1.8 – Schéma du modèle de calcul de CTA extrait du *Technical Design Report DATA* de 2015. Ce modèle distribué prévoit un nombre limité de centres de calcul, 4 en l'occurence, se répartissant le traitement et l'archivage des données. Un des 4 centres assurera également toutes les activités de gestion (orchestration du traitement et de l'archivage, gestion des propositions d'observation, etc.) et sera l'unique interface pour les utilisateurs de l'observatoire (*Gateway Services*).

Afin de minimiser les temps de latence dus aux transferts sur le réseau, le traitement des données sera distribué entre les 4 centres de préférence en fonction de la localisation des données. L'orchestration de la réplication sera assurée par le système d'archivage, alors que l'exécution des différents pipelines sur les 4 centres sera gérée par le gestionnaire de tâches ou `CWMS` (*Computing Workload Management System*). Ces deux composants, présentés plus en détail respectivement dans les Sections 1.4.4 et 2.7, seront opérés par l'un des 4 centres, baptisé *Science Data Management Center*. Ce dernier sera en charge de toutes les activités de gestion et hébergera également le portail scientifique, unique point d'accès pour les utilisateurs de l'observatoire (*Science Gateway*).

Pour l'implémentation de ce modèle de calcul, CTAO établira des accords avec des centres

de calcul de renommée internationale, possédant les ressources et l'expertise dans le traitement et le stockage de grandes quantités de données, tels les Tier-1 ou les Tier-2 de WLCG. En ce qui concerne, le système d'archivage et le gestionnaire de tâches `CWMS`, plusieurs prototypes ont été développés au sein du consortium. Dans le Chapitre 3, je vais présenter en détail le prototype que j'ai développé pour le gestionnaire des tâches.

1.4.4 Le système d'archivage

La Figure 1.9 représente schématiquement le flux de données depuis les caméras jusqu'aux utilisateurs finaux. Les principaux composants qui interviennent dans l'archivage et le traitement des données y sont également représentés. Le système d'archivage est un système logiciel dont les fonctions principales sont :

- Garantir la préservation à long terme des données de l'observatoire
- Garantir un accès efficace aux données au gestionnaire des tâches et aux utilisateurs
- Fournir une interface permettant de sélectionner les données à travers des requêtes sur des méta-données

Il est à noter que le système d'archivage ne comprend pas les ressources de stockage, mais il fournit des interfaces de haut niveau pour un accès transparent aux données hébergées dans les centres de calcul. Le système d'archivage est en charge de l'orchestration de la réplication et des transferts des données entre les différents centres selon le modèle de calcul décrit dans la Section 1.4.3.

D'un point de vue fonctionnel on distingue deux systèmes d'archivage ayant les mêmes fonctionnalités de base mais des cas d'utilisation et des contraintes différentes : l'archive 'bulk' et l'archive 'scientifique'. L'archive bulk gère les données de bas niveau (DL0-DL3), alors que l'archive scientifique est dédiée aux données de haut niveau (DL3-DL5). De manière analogue, on distingue également deux activités de traitement de données (cf. Figure 1.9) : l'activité de *Final Processing* concerne le traitement des données du niveau DL0 au niveau DL3, alors que l'activité de *Science Data Preparation* concerne le traitement du niveau DL3 au niveau DL5. À ces deux activités de traitement s'ajoute l'activité de *Simulation*, également représentée sur la Figure 1.9. L'archive bulk gère un grand volume de données de l'ordre de dizaines de Po par an, alors que l'archive scientifique gère une quantité de données de l'ordre de quelques dizaines de Go. Les modalités d'accès aux deux archives sont également très différentes. L'archive bulk est consultée essentiellement par le gestionnaire des tâches `CWMS` afin d'exécuter les pipelines pour le *Final Processing* et les *Simulations*. L'accès est restreint au personnel de CTAO, mais il doit être particulièrement efficace, notamment car il est prévu de re-traiter toutes les données archivées au moins une fois par an.

En revanche, l'archive scientifique a des contraintes moindres en termes de débit d'accès pour la *Science Data Preparation*, mais un grand nombre d'utilisateurs y accède, les données de haut niveau étant publiques à la fin de la période propriétaire. La gestion des droits d'accès est donc particulièrement délicate. En outre, les données de haut niveau doivent être conformes aux standards de l'Observatoire Virtuel. L'accès par le public se fera uniquement à travers un portail web (*Science Portal*), via les outils et les protocoles de l'Observatoire Virtuel. Du point de vue de l'implémentation, les deux archives pourront éventuellement utiliser la même solution logicielle et ou partager la même infrastructure. Plusieurs systèmes

logiciels sont actuellement en cours d'évaluation ou de prototypage, tels DIRAC, Rucio [8] et Onedata [4].

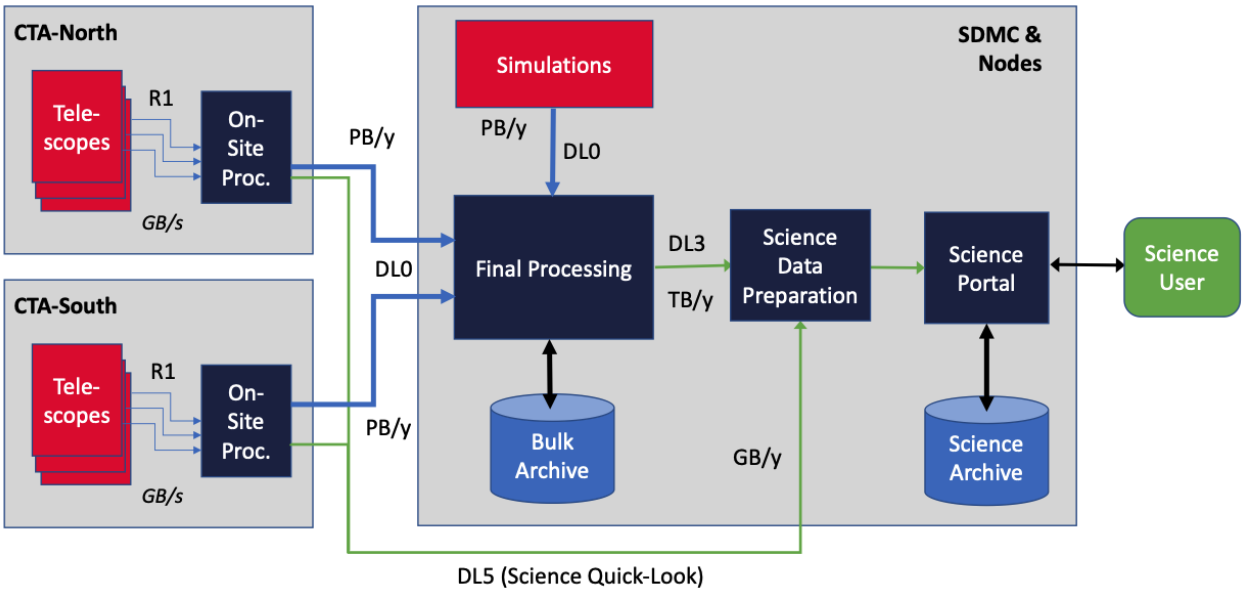

FIGURE 1.9 – Schéma du flux de données à travers les principales étapes de traitement sur site et hors ligne.

1.4.5 Le gestionnaire des tâches de calcul

Le gestionnaire des tâches (`CWMS`) est le composant logiciel en charge de l'orchestration des tâches pour l'exécution des pipelines de traitement et de simulation décrits dans la Section 1.4.1. Il s'agit de la couche logicielle entre les utilisateurs des pipelines et l'infrastructure de calcul. Le gestionnaire des tâches permet en particulier d'implémenter le modèle de calcul présenté dans la Section 1.4.3. Les fonctionnalités principales requises sont les suivantes :

- Fournir une interface transparente pour l'accès à des ressources de calcul distribuées et hétérogènes
- Optimiser l'utilisation des ressources de calcul à travers un ordonnancement efficace
- Orchestrer l'exécution des pipelines sur l'infrastructure de calcul

Sur le schéma de la Figure 1.9, le gestionnaire des tâches est en charge des activités de *Final Processing*, *Science Data Preparation* et *Simulations*. Ces différentes activités imposent des contraintes différentes. Les simulations nécessitent un nombre important de CPUs avec des éventuels pics d'utilisation. Le *Final Processing* comporte d'une part une utilisation modérée des ressources CPUs lissée sur l'année pour le traitement des données au fil des observations et d'autre part une utilisation intensive lors des campagnes de re-traitement.

4. `https://onedata.org/docs/doc/using_onedata/file_management.html`

Enfin, les traitements liés à l'activité de *Science Data Preparation* sont beaucoup plus rapides et consomment peu de CPU.

Le `CWMS` doit donc supporter toutes ces activités concurrentes dans les délais impartis. Malgré les différentes contraintes liées à ces activités, la solution logicielle pour le `CWMS` pourrait être partagée. L'expérience montre que dans un contexte d'utilisation d'un grand nombre de ressources distribuées, le risque de ressources défaillantes est élevé. Le gestionnaire de tâches doit donc implémenter un mécanisme de récupération des échecs très efficace. Une autre caractéristique essentielle du `CWMS` consiste à permettre d'intégrer facilement de nouveaux types de ressources que l'observatoire voudrait éventuellement utiliser, comme par exemple des ressources commerciales de type *cloud*.

Dans les Chapitres 2 et 3 je vais présenter de manière détaillée le framework DIRAC et le prototype de gestionnaire de tâches que j'ai développé pour CTA sur la base de ce framework (projet `CTA-DIRAC`). Les activités que j'ai ciblées plus particulièrement pour l'évaluation de DIRAC sont le *Final Processing* et les *Simulations*. Néanmoins dans le cadre du projet `CTA-DIRAC`, j'ai également utilisé de manière intensive ses fonctionnalités de gestion de données et donc de système d'archivage.

Chapitre 2

Gestion du calcul dans un environnement distribué avec DIRAC

2.1 Contexte

DIRAC est un logiciel pour la gestion des tâches de calcul (WMS) et des données (DMS) dans un environnement de calcul distribué. Il est conçu comme une couche logicielle entre les ressources de calcul et les utilisateurs, comme illustré sur la Figure 2.1. Il a été développé

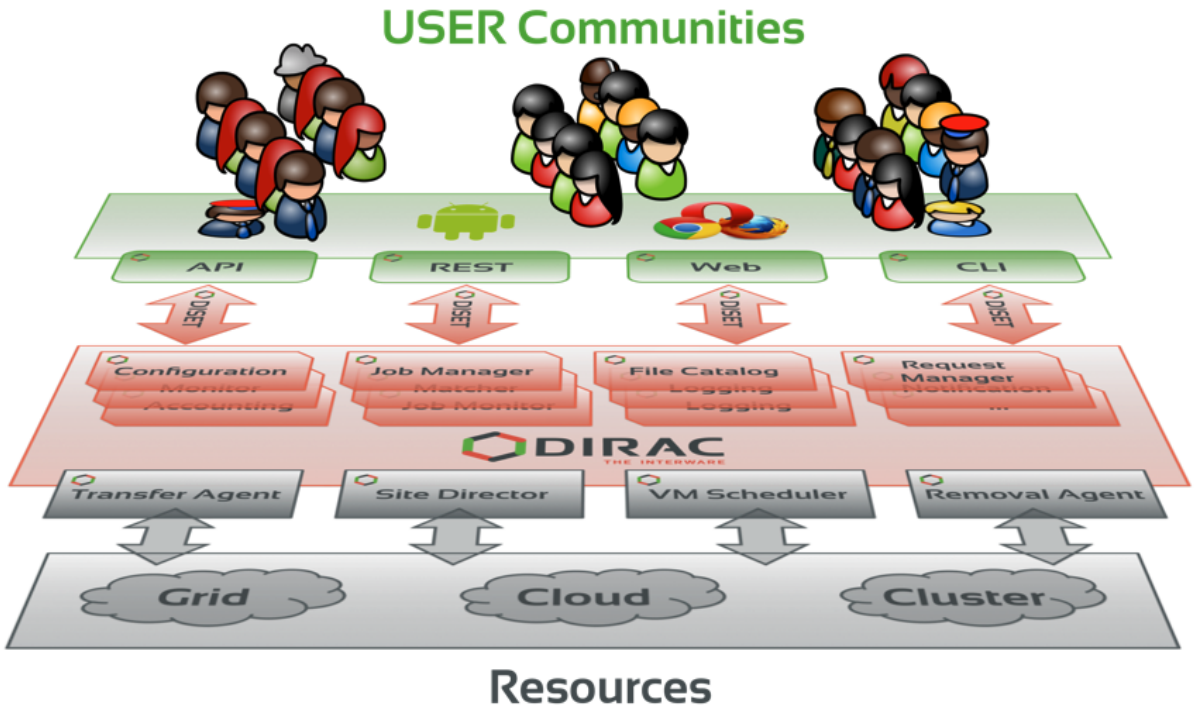

FIGURE 2.1 – DIRAC est une couche logicielle entre les communautés d'utilisateurs (en haut) et des ressources de calcul distribuées et hétérogènes (en bas).

à l'origine par l'expérience LHCb au CERN au début des années 2000 pour supporter les différentes activités de simulation et de traitement de données sur la grille WLCG. En effet, les premiers WMS développés dans le cadre des projets Data Grid et EGEE présentaient quelques limitations en terme de taux de réussite des tâches et de capacité à occuper les ressources disponibles. Ce constat a conduit les quatre expériences LHC à développer leur propre WMS (DIRAC pour LHCb, Panda pour ATLAS, glideInWMS pour CMS et Alien pour ALICE). Parmi ces systèmes, DIRAC fut l'un des premiers à introduire un ordonnancement basé sur un paradigme de type *pull* en opposition avec le type *push* implémenté par ses prédécesseurs [9] [10]. L'ordonnancement de type *pull* est implémenté par un mécanisme nommé *pilot job* (ou *late binding*). D'un point de vue formel, une explication théorique de l'efficacité d'un ordonnancement de type *pull* est donnée dans [11]. Dans la Section 2.7 je vais présenter comment ce mécanisme est implémenté dans DIRAC. Les travaux dans [12] montrent, à l'aide de simulations de systèmes distribués, que l'ordonnancement implémenté dans DIRAC est plus efficace par rapport à des systèmes de type *push*.

2.2 Le Consortium DIRAC

Au delà du LHC, de nombreuses expériences appartenant à différents domaines scientifiques se sont retrouvées confrontées à la même problématique d'exploiter efficacement des ressources distribuées à grande échelle. À travers le projet `CTA-DIRAC`, CTA a été l'une des premières communautés à adopter DIRAC en dehors de LHCb. Depuis, plusieurs autres communautés ont suivi. Aujourd'hui les principales communautés d'utilisateurs de DIRAC sont : LHCb, CTA, ILC, Belle II, Bes III, Pierre Auger et Juno.

C'est dans ce contexte qu'en 2014, un Consortium a été constitué autour de DIRAC, associant 3 partenaires fondateurs : le CNRS au travers du CPPM, le CERN et l'Université de Barcelone. L'objectif du Consortium DIRAC est de formaliser la collaboration entre les partenaires pour assurer le développement et la maintenance de DIRAC (*Agreement for Development and Maintenance of the DIRAC Software for Distributed Computing*).

En 2015, du fait de ma forte implication dans le projet `CTA-DIRAC`, l'Université de Montpellier, en tant que tutelle du LUPM, a été invitée à rejoindre formellement le Consortium. Aujourd'hui ce dernier compte plusieurs autres partenaires, notamment KEK, IHEP, PNNL et Imperial College. Les membres du Consortium détiennent les droits d'auteurs du logiciel DIRAC [1], qui est distribué sous la licence libre GPL V3. Le Consortium est gouverné par un *comité de représentants*, qui est responsable des *releases* du logiciel et qui décide des axes de développement à moyen et à long terme. Depuis 2015, je participe donc aux réunions du comité, en tant que représentante de l'Université de Montpellier, dans lesquelles nous décidons les orientations du projet.

Chaque année, nous organisons également un atelier d'une durée d'une semaine (*DIRAC User Workshop*), rassemblant les développeurs et les communautés d'utilisateurs de DIRAC. Ces ateliers sont organisés à tour de rôle par les partenaires du Consortium et en 2016 j'ai pris en charge l'organisation de la sixième édition à Montpellier [2]. Les développeurs y présentent le status de DIRAC ainsi que les évolutions prévues et les utilisateurs leur cas

1. `https://github.com/DIRACGrid/DIRAC`
2. `https://indico.cern.ch/event/477578/overview`

d'utilisation en exprimant également les besoins pour d'éventuelles nouvelles fonctionnalités. Lors de ces ateliers, nous organisons également des sessions de formation pour les utilisateurs, les développeurs et les administrateurs de services DIRAC. Les échanges sont extrêmement riches et bénéfiques pour l'ensemble des participants.

Au niveau français j'ai également participé à la création du projet `DIRAC@IN2P3`, présenté dans l'Annexe A.1, ayant le but de fédérer la communauté française autour de DIRAC et de soutenir l'effort de généralisation du logiciel.

2.3 Composants principaux

DIRAC, écrit en python, est conçu selon une architecture modulaire, où chaque module, appelé 'système', expose des fonctionnalités de haut niveau pour réaliser un ensemble de tâches spécifiques. Les principaux systèmes de DIRAC sont :

- *Configuration System* (CS) : gestion de la configuration de DIRAC
- *Workload Management System* (WMS) : gestion des tâches de calcul
- *Data Management System* (DMS) : gestion des données
- *Transformation* et *Production Systems* (TS, PS) : gestion des *workflows* de haut niveau
- *Request Management System* (RMS) : gestion asynchrone de requêtes

D'autres systèmes complètent cet ensemble en fournissant des fonctionnalités supplémentaires :

- *Ressource Status System*
- *Monitoring System*
- *Accounting System*
- *Framework System*

Tous ces systèmes interagissent entre eux à travers des communications client/server. Grâce à la modularité de DIRAC, il est possible d'utiliser seulement certains de ces systèmes en fonction du cas d'application. Dans les sections suivantes, je vais présenter les concepts généraux des systèmes principaux ainsi que mes contributions aux systèmes de gestion des *workflows*.

2.4 Architecture logicielle

L'architecture logicielle de DIRAC est une architecture orientée service (*Service Oriented Architecture*). Dans cette architecture, chaque système est constitué des composants suivants :

- Bases de données (DB) : elles conservent l'état du système et jouent le rôle d'une sorte de mémoire partagée pour le système
- Services : ce sont des composants passifs qui écoutent les requêtes venant des clients. Ils renvoient les informations demandées en interrogeant la DB associée ou bien ils insèrent des données dans la DB. Les Services peuvent être à leur tour clients d'autres Services appartenant au même système ou à d'autres systèmes

- Agents : ce sont des composants actifs dont l'exécution est invoquée à des intervalles de temps réguliers, de manière similaire à des *cron jobs*. À chaque cycle les Agents exécutent la même suite d'instructions en formulant des requêtes aux Services DIRAC ou à des services tiers
- Executors : ce sont également des composants actifs dont l'exécution est invoquée sur demande, de manière similaire aux *consumers* d'un système de *Message Queuing*. Ils sont utilisés en particulier dans le WMS

Les clients n'interagissent donc jamais directement avec les bases de données, mais toujours à travers un Service, selon le flux de communication suivant : client → Service → DB. Les communications client/Service se font à travers des appels RPC, utilisant un protocole sécurisé propre à DIRAC (DISET), développé comme une couche logicielle au dessus d'OpenSSL. Pour le mécanisme d'authentification, le protocole DISET utilise les certificats X509. Tous ces composants sont installés sur un ou plusieurs serveurs dédiés. Plusieurs types d'installations de DIRAC sont disponibles en fonction du contexte d'utilisation :

- Installation serveur : elle comprend tous les composants de DIRAC, y compris les Services et les DBs. Les différents composants peuvent être installés sur un ou plusieurs serveurs, éventuellement distribués géographiquement
- Installation cliente : installation légère, qui comprend essentiellement les interfaces API et ligne de commande. Les utilisateurs ont ce type d'installation sur leur poste de travail
- Installation *pilot* : installation de type client effectuée par les *pilot jobs* sur les noeuds de calcul

2.5 Le Système de Configuration

Chaque installation server (ou instance) de DIRAC est configurée de manière spécifique en fonction des besoins de la communauté qui l'utilise. Cette configuration contient des données statiques (ou semi-statiques) nécessaires au fonctionnement de l'instance. Elle est organisée en plusieurs sections principales selon une structure hiérarchique. Elle contient notamment : la description des ressources de calcul et de stockage auxquelles la communauté a accès ; un registre des utilisateurs de l'instance organisé en groupes ayant différents niveaux d'autorisation ; les paramètres de tous les composants de l'instance (Agents/Executors, Services, DB) ainsi que les *endpoints* des Services. Enfin, une section est dédiée à des paramètres opérationnels qui sont ajustés régulièrement en fonction du modèle de calcul de la communauté d'utilisateurs, e.g. la politique d'accès aux données de la part des tâches, les sites éligibles à exécuter certains types de tâches, etc.

Cette configuration est mise à jour et consultée à travers un Service spécifique, appelé *Configuration Service*. Ce Service comprend une instance *Master*, ayant un accès en lecture/écriture à la configuration permettant une mise à jour des différentes sections, et une ou plusieurs instances *Slave* ayant un accès en lecture seule. La disponibilité de ce Service étant critique pour le fonctionnement d'une instance, on installe typiquement plusieurs instances *Slaves* sur différents serveurs de préférence distribués géographiquement.

2.6 Les interfaces

Différentes types d'interfaces permettent aux utilisateurs d'interagir avec les Services DIRAC :

- La ligne de commande
- L'API python
- L'interface REST (*REpresentational State Transfer*) [13]
- Le portail web

À travers ces différentes interfaces, les utilisateurs peuvent soumettre leurs tâches de calcul, les monitorer, récupérer les résultats et effectuer des opérations de gestion de données. Typiquement, ils utilisent la ligne de commande depuis leur installation cliente ainsi que le portail web par exemple pour le monitorage des tâches. L'API python est particulièrement pratique pour les utilisateurs avancés qui souhaitent programmer des fonctionnalités spécifiques. Les différents Services interagissent également entre eux en utilisant les APIs correspondantes. Enfin, l'interface REST est plutôt dédiée à des services tiers, comme par exemple les portails scientifiques (*science gateways*). Généralement, les utilisateurs configurent leurs *workflows* sur ces portails, qui ensuite utilisent l'interface REST pour envoyer les tâches à DIRAC.

2.7 La gestion des tâches, WMS

La gestion des tâches dans DIRAC est assurée par le *Workload Management System*. Le WMS gère tout le cycle de vie des tâches, à partir de leur soumission jusqu'à leur exécution et à la récupération des résultats de la part des utilisateurs. L'un des éléments clés du WMS de DIRAC est l'implémentation du mécanisme du *pilot job*.

La Figure 2.2 illustre les quatre étapes principales et les composants DIRAC qui interviennent dans l'implémentation du *pilot job* : **1)** soumission des tâches (ou *payloads*) de la part des utilisateurs ; **2)** réservation des ressources ; **3)** exécution des tâches ; **4)** récupération des résultats de la part des utilisateurs.

Les utilisateurs préparent une description générique de la tâche et la soumettent au WMS à travers le Service *Job Manager*, qui l'insère dans la *Job DB* (**soumission des tâches**). Une série d'Executors (*Job Sanity*, *Job Path*, etc.) est ensuite invoquée par le Service d'*Optimization Mind*. Ils sont en charge des différentes étapes de vérification et ordonnancement de la tâche en fonction de ses spécifications et de la localisation des données d'entrée. Suite à cette phase, la tâche est associée à une *TaskQueue* (et sauvegardée dans la *TaskQueue DB*) qui regroupe toutes les tâches ayant les mêmes spécifications et pouvant être exécutées sur le même type de ressource.

Des Agents dédiés, nommés *Pilot Schedulers*, vérifient en permanence le statut de la *TaskQueue DB*. Les différents *Pilot Schedulers* sont en charge de la soumission des *pilot jobs* et chacun est spécialisé pour un type particulier de ressource. Lorsqu'un *Pilot Scheduler* trouve dans la *TaskQueue DB* des tâches destinées aux ressources pour lesquelles il est spécialisé, il réserve une ressource en envoyant un *pilot job* (**réservation des ressources**). Dans la suite les tâches des utilisateurs sont également appelées *payloads* pour les distinguer des *pilot jobs*.

Le *pilot job* est une tâche générique qui vérifie l'état de la ressource sur laquelle il s'exécute et prépare l'environnement d'exécution du *payload*. Après ces vérifications, le *pilot job* contacte le Service *Matcher* auquel il présente la description de la ressource. Le *Matcher* vérifie si un ou plusieurs *payloads* dans la *TaskQueue DB* ont des spécifications correspondant à cette ressource. Si plusieurs *payloads* satisfont les critères du *Matcher*, le *payload* le plus prioritaire sera sélectionné pour exécution (**exécution des tâches**). On parle d'un paradigme de *pull* ou *late binding*. Si le Service de *Matcher* ne trouve aucune correspondance, alors le *pilot job* termine son exécution sans erreurs.

Les avantages de ce mécanisme sont multiples. Tout d'abord, si le *pilot job* termine en erreur, car par exemple la ressource sur laquelle il s'exécute est défaillante, il ne récupérera aucun *payload* pour exécution. Le taux d'échec des *payloads* est ainsi minimisé.

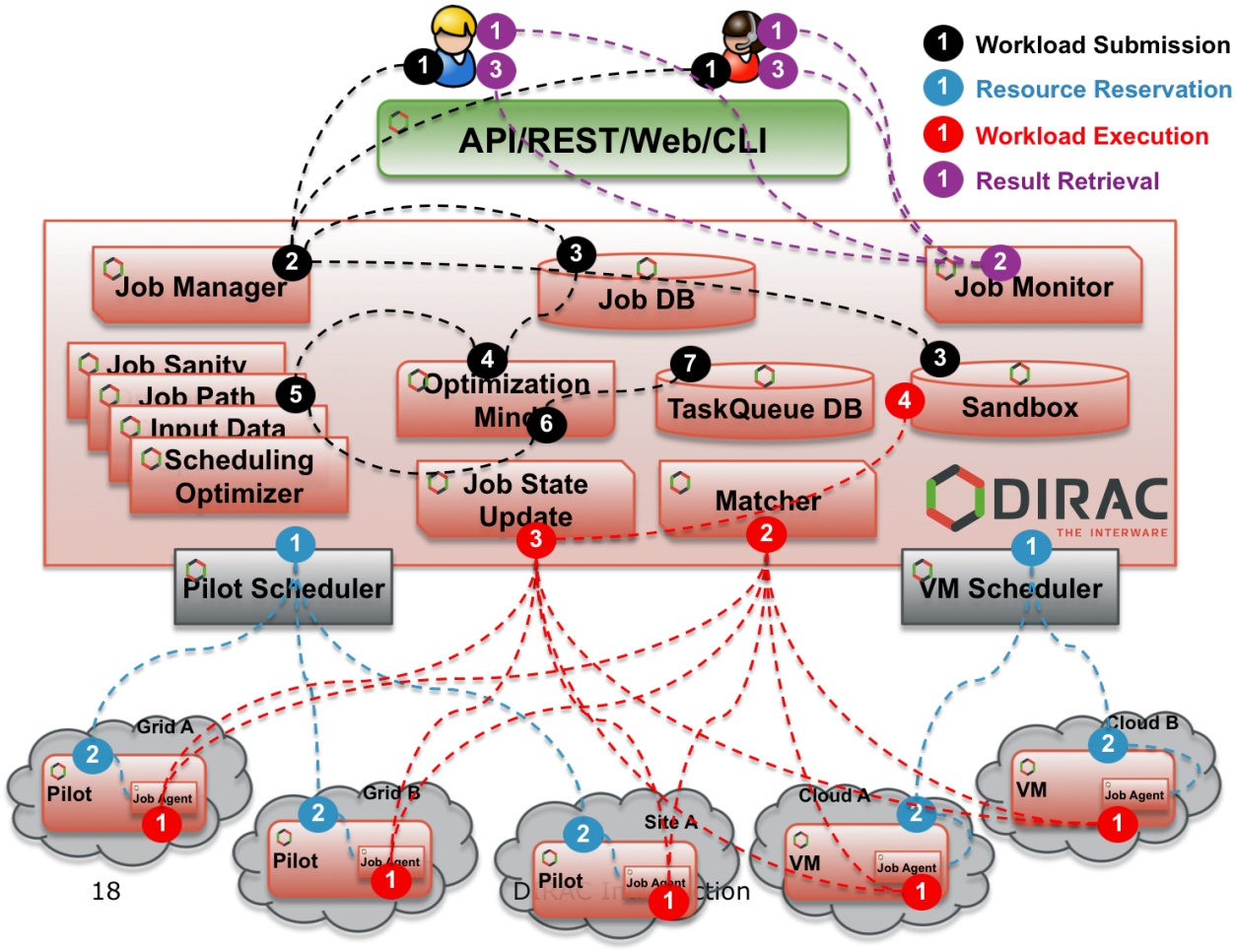

FIGURE 2.2 – Vue des différents composants du WMS de DIRAC qui interviennent dans l'implémentation du *pilot job*. Les étapes principales du cycle de vie d'une tâche sont représentées : **1)** soumission des tâches de la part des utilisateurs (en noir) ; **2)** réservation des ressources (en bleu) ; **3)** exécution des tâches (en rouge) ; **4)** récupération des résultats (en violet).

Deuxièmement ce mécanisme permet de réserver les ressources et donc de réduire le délai entre la soumission et l'exécution du *payload*. Puisque chaque *pilot job* peut récupérer le *payload* de n'importe quel utilisateur, dès qu'un *pilot job* trouve une correspondance avec un *payload*, ce dernier sera exécuté. De plus, lorsque l'exécution d'un *payload* est terminée, un autre *payload* est immédiatement récupéré pour exécution et ainsi de suite jusqu'à ce que la limite CPU de la ressource soit atteinte (*filling mode*). Ceci a également pour effet de réduire considérablement la surcharge due à la soumission des tâches.

Un autre avantage réside dans le fait que la description de la ressource présentée par le *pilot job* est une description réaliste, ce qui minimise les échecs dus à une description erronée. Dans les systèmes d'ordonnancement de type *push*, les ressources sont sélectionnées sur la base de la correspondance entre les spécifications du *payload* et les capacités des ressources publiées dans un système d'information (BDII pour le *middleware* gLite). Le problème de cette approche est que dans un environnement fortement dynamique comme celui de la grille, bien souvent les informations publiées ne sont pas correctes.

Une des premières démonstrations pratiques de l'efficacité du mécanisme du *pilot job* de DIRAC fut obtenue par l'expérience LHCb lors du *Data Challenge* de 2004, qui mesura un taux de succès des tâches supérieur à 90% à comparer avec un taux de 60% obtenu avec le système précédent (LCG Resource Broker) [14].

2.7.1 Intégration de ressources

Comme mentionné dans la section précédente, les Agents du WMS en charge de la soumission des *pilot job* (*Pilot Schedulers*) sont chacun spécialisés pour un type particulier de ressources. De cette manière, le WMS de DIRAC est capable d'agréger différents types de ressources, de manière totalement transparente pour l'utilisateur.

2.7.1.1 Ressources de type HTC (*High Throughput Computing*)

DIRAC a été développé à l'origine pour cibler spécialement des ressources HTC de type de grille, qui sont encore aujourd'hui les plus utilisées par la communauté de physique des hautes énergies. Ces ressources sont accessibles à travers des services de *Computing Element*, qui exposent une interface standard pour la soumission et le monitorage de tâches gérées par différents systèmes de batch. Plusieurs services de *Computing Element* ont été développés au cours des dernières années dans différents projets de *middleware*. DIRAC a suivi ces évolutions et aujourd'hui permet de soumettre des tâches à plusieurs types de *Computing Elements* (CREAM, HTCondorCE, ARC) à travers des *Pilot Schedulers* spécialisés.

En outre, DIRAC permet d'accéder également à des ressources dans des centres qui n'appartiennent pas une grille de calcul et qui ne fournissent pas un service d'accès distant. Dans ce cas, DIRAC accède au système de batch du centre en question à travers un tunnel ssh, tel qu'illustré sur la Figure 2.3. Les systèmes de batch actuellement supportés sont : Grid Engine, LSF, OAR, Slurm, Condor, Torque. Cette possibilité offerte par DIRAC permet aux différentes institutions de mettre à disposition très facilement des ressources à leurs utilisateurs, sans la surcharge du déploiement d'un site grille.

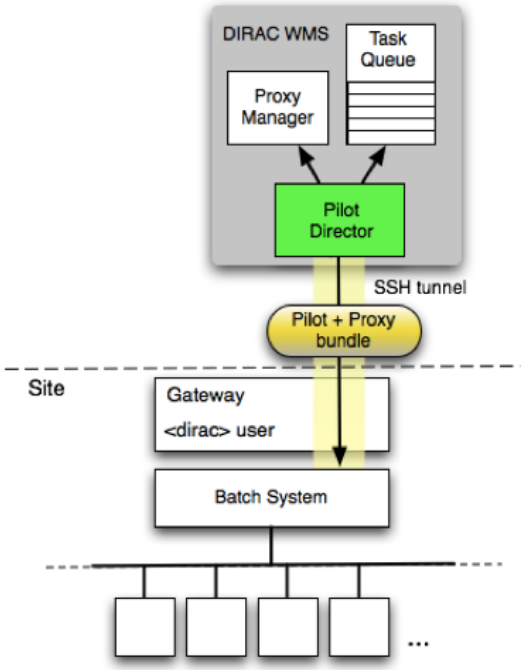

FIGURE 2.3 – Schéma montrant la soumission de tâches sur des fermes de calcul ne disposant pas d'un service d'accès distant via le WMS de DIRAC. Un Agent dédié soumet des *pilots jobs* au système de batch à travers un tunnel ssh.

2.7.1.2 Ressources de type HPC (*High Performance Computing*)

Les centres HPC sont de plus en plus utilisés par les communautés de physique des hautes énergies et d'astrophysique [15]. Bien que les supercalculateurs dans ces centres soient plutôt adaptés à des tâches *multi-core* ou *multi-node*, les centres HPC mettent de plus en plus à disposition leurs ressources à des tâches *single-core* pour assurer un taux d'occupation maximal. De plus, l'utilisation de GPUs, dont les centres HPC sont souvent équipés, est de plus en plus répandue dans les communautés HEP en particulier pour des applications d'apprentissage profond. Un exemple d'utilisation des GPUs pour l'imagerie médicale via DIRAC est donné dans [16].

La difficulté majeure de l'intégration de ces ressources de la part du WMS est liée à des politiques de sécurité assez strictes pour l'accès aux centres HPC. Comme illustré sur la Figure 2.4, les noeuds de calcul n'ont pas généralement de connectivité réseau vers l'extérieur, ce qui empêche les *pilot jobs* de communiquer avec les services DIRAC situés en dehors des centres. La solution proposée par DIRAC pour l'accès à ces ressources consiste dans le déploiement d'un Service de *Gateway* sur une machine du centre ayant accès à la fois au réseau interne et au réseau externe (machine de *login*).

Une complexité ultérieure s'ajoute lorsque les tâches nécessitent d'accéder à des fichiers stockés sur des sites distants. Les fichiers doivent alors transiter dans une zone tampon sur la machine de *login*, ce qui requiert le déploiement de composants DIRAC supplémentaires.

Enfin, bien que la plupart des applications HEP soient de type *single-core*, certaines applications sont conçues pour s'exécuter sur plusieurs coeurs ou plusieurs noeuds. Le WMS de DIRAC supporte les applications *multi-core* en essayant de maximiser l'utilisation des ressources. Le principe de fonctionnement, illustré sur la Figure 2.4, consiste à déployer des *pilot jobs* avec des *slots* internes partitionnables. Plus précisément, des *pilot job* M-core récupèrent des applications N-core (où N <= M) à travers le même mécanisme de *pull* décrit dans la Section 2.7, jusqu'à remplir tous les coeurs d'un CPU.

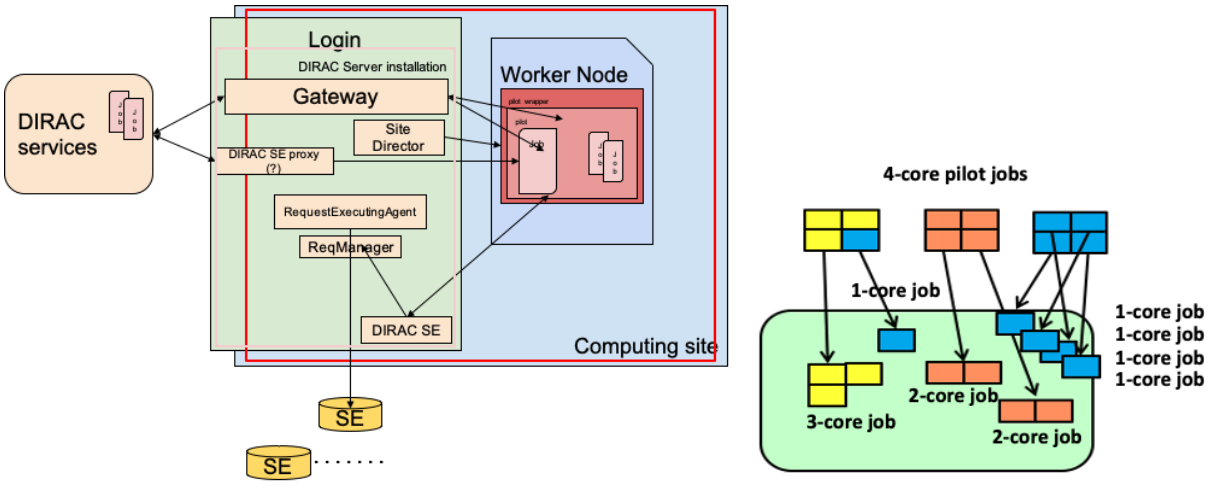

FIGURE 2.4 – À gauche : schéma du fonctionnement de DIRAC pour accéder aux ressources dans un centre HPC n'ayant pas de connectivité réseau vers l'extérieur. À droite : des *pilot jobs* 4-core sont déployés et exécutent des applications N-core (N=1,2,3) en maximisant l'utilisation des coeurs du CPU.

2.7.1.3 Ressources cloud

Face aux besoins croissants en calcul de la communauté scientifique, plusieurs collaborations ont commencé à utiliser des clouds commerciaux (google, amazon, etc.) comme moyen de disposer rapidement de ressources supplémentaires. Parmi les avantages principaux offerts par le cloud et qui représentent un intérêt pour les applications scientifiques, on mentionne :

- La possibilité pour les utilisateurs de disposer directement des ressources sans réservation préalable (*on-demande self-service*)
- La capacité d'allouer rapidement des ressources pour répondre aux pics de demande (*rapid elasticity*)
- La possibilité de personnaliser l'environnement d'exécution en utilisant des machines virtuelles

Les principales applications dans la communauté HEP consistent à utiliser le cloud en complément des ressources grille pour absorber les pics des simulations Monte-Carlo. Les simulations demandent en effet une puissance de calcul élevée, mais comportent peu de transferts de données. En revanche, l'utilisation du cloud pour le traitement de données à grande échelle pose des défis supplémentaires, car le cloud ne dispose pas d'un service de stockage distribué et fédéré comparable à celui de la grille.

Un autre exemple d'utilisation vient des expériences LHC, qui ont transformé en cloud la ferme *on-line* dédiée au système de déclenchement (*High Level trigger*). L'objectif est celui d'exploiter au maximum la puissance de calcul de cette ferme pendant les périodes d'arrêt du LHC. L'élasticité du cloud permet en effet de transformer rapidement l'utilisation de cette ferme pour des activités *off-line*.

La facilité de déploiement et d'administration d'un cloud par rapport à un site grille a également contribué à son essor dans nos instituts. Au delà de l'utilisation du cloud pour le calcul, les administrateurs ont progressivement migré sur le cloud un certain nombre de services destinés aux utilisateurs locaux, ce qui facilite l'administration globale.

Malgré ces différents avantages, l'utilisation du cloud de la part des communautés scientifiques présente un certain nombre de défis. Pendant longtemps, les changements fréquents des interfaces des différents *Cloud Managers* (OpenStack, OpenNebula, etc.) et le manque d'un standard pour ces interfaces ont constitué un obstacle à leur intégration stable dans les différents WMS. L'accès à des infrastructures de cloud distribuées souffre également du manque d'un système d'authentification basé sur des fédérations d'identité. Enfin, la gestion de la répartition des ressources entre plusieurs groupes d'utilisateurs (*fair-sharing*) sur le cloud est relativement limitée.

Afin de relever ces défis et de faciliter l'utilisation des clouds commerciaux et institutionnels de la part des communautés scientifiques, plusieurs projets ont été financés par la communauté européenne. Le projet INDIGODataCloud [17] (2015-2017) a développé différents composants *middleware* précisément pour répondre à ces défis.

Le projet EGI Federated Cloud [3] a développé une fédération de clouds institutionnels. L'idée de base consiste à offrir un service de cloud IaaS (*Infrastructure as a Service*) tout en reprenant le modèle de la grille d'une fédération de ressources.

3. https ://www.egi.eu/federation/egi-federated-cloud/

Enfin, le projet HNSciCloud (2016-2018) [4] a exploré la viabilité d'un modèle de cloud hybride public-privé, le résultat final étant une feuille de route pour l'implémentation d'un tel modèle [18]. J'ai personnellement participé à ce projet en tant que représentante de CTA, en intégrant les clouds publics et privés fournis par les partenaires dans l'environnement de production de CTA (prototype `CTA-DIRAC`, cf. Section 3).

Dans ce contexte, les différents projets de WMS ont progressivement intégré le support des ressources clouds. DIRAC dispose d'un module, spécialement dédié à la gestion des tâches sur le cloud (VMDIRAC), qui implémente le même mécanisme de *pilot job* utilisé pour les ressources HTC, bien qu'il présente quelques spécificités. La Figure 2.5 illustre son principe de fonctionnement.

Le *VM Scheduler* est un service qui orchestre la création et l'arrêt des machines virtuelles en fonction du statut de la *TaskQueue*. Il convient de noter qu'il s'agit de la même *TaskQueue* qui est consultée par les *Pilot Schedulers* pour la soumission sur des ressources HTC et HPC. Ceci permet donc d'optimiser l'ordonnancement en prenant en compte le statut global de toutes les ressources. Du point de vue de l'utilisateur la description des tâches et l'interface de soumission restent identiques. La création des machines virtuelles est ensuite effectuée par des Agents, *Cloud Directors*, de manière analogue aux *Pilot Schedulers*. Plusieurs *middleware* clouds sont actuellement disponibles sur le marché, e.g. OpenStack, OpenNebula, etc. Chaque *Cloud Director* est donc spécialisé pour un *middleware* cloud particulier. DIRAC supporte actuellement OpenStack, OpenNebula, EC2 ainsi que l'interface rOCCI. La contextualisation des machines virtuelles est réalisée à travers un mécanisme de *bootstrapping* qui permet d'exécuter les mêmes types de *pilots jobs* utilisés sur les sites grille.

Une alternative intéressante pour la gestion des tâches sur le cloud est représentée par le modèle Vacuum [19], dans lequel la gestion des machines virtuelles est déléguée aux fournisseurs de ressources. Dans ce modèle, les machines virtuelles sont instanciées et configurées par les fournisseurs de ressources, alors que la configuration de la contextualisation sont fournies par les utilisateurs. L'avantage de ce modèle est que les utilisateurs n'ont pas à gérer l'instanciation des machines virtuelles et donc le *middleware* cloud. La récupération des *payloads* en revanche suit le même mécanisme que les *pilot jobs* par le biais d'un Agent exécuté sur la machine virtuelle. Plusieurs implémentations de ce modèle existent (Vac, Vcycle [20] et HTCondor Vacuum) et ont été adoptées par différents sites cloud. Pour les utilisateurs ayant accès à des sites gérés selon le modèle Vacuum, l'exploitation des ressources s'est avérée très efficace [21] [22].

2.8 La gestion des données

Les différentes tâches qui composent un *workflow* produisent des données qui doivent être stockées de manière permanente, éventuellement transférées, accessibles par d'autres tâches et référencées dans un catalogue.

Comme nous l'avons vu dans la Section 2.3, DIRAC comprend également un système de gestion de données (*Data Management System* ou DMS). Bien que l'architecture modulaire de DIRAC permette d'utiliser indépendamment le WMS et le DMS, les interactions entre ces deux systèmes sont multiples. Afin de distribuer les tâches selon un modèle de calcul défini

4. https ://www.hnscicloud.eu/

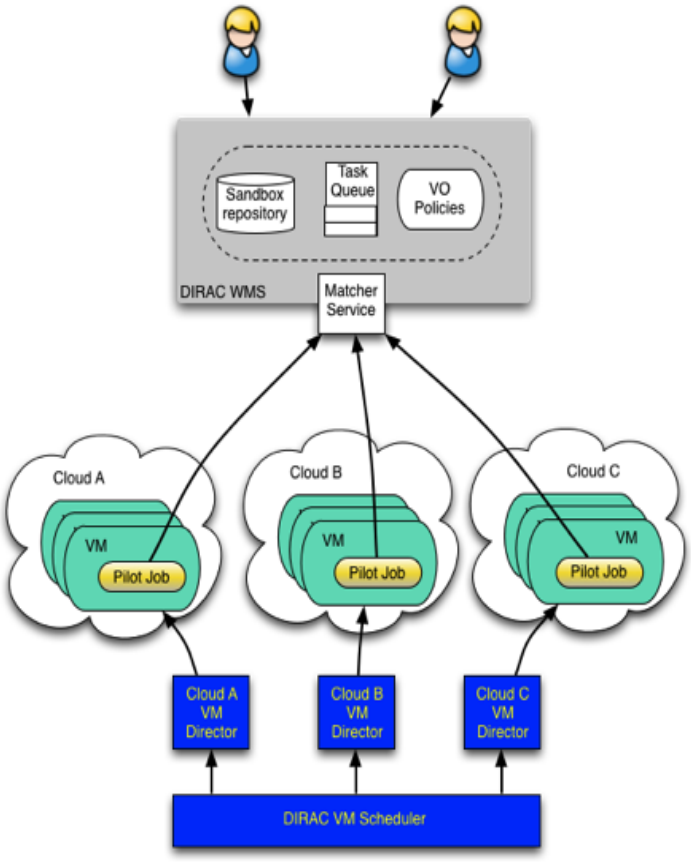

FIGURE 2.5 – Schéma du fonctionnement du module VMDIRAC pour la gestion des machines virtuelles sur le cloud. Le *VM Scheduler* orchestre la création et l'arrêt des machines virtuelles en fonction du statut de la *TaskQueue*. L'instanciation des machines virtuelles est assurée par des Agents spécialisés pour chaque *middleware* cloud (e.g. Cloud A, B, C). Les machines virtuelles démarrent des *pilots jobs* identiques à ceux utilisés sur les sites grille. Ces derniers contactent ensuite le service *Matcher* pour récupérer des tâches à exécuter.

(par exemple en exécutant les tâches au plus proche des données), le WMS doit connaître la localisation physique des données. Dans la Section 2.10.1, nous verrons également que dans DIRAC l'enchainement des étapes d'un *workflow* est complètement guidé par les données et leurs métadonnées. Ceci implique que le gestionnaire de *workflow* interroge massivement le catalogue de fichiers et de métadonnées. Dans cette section je vais présenter les principaux concepts du *Data Management System* de DIRAC.

2.8.1 Intégration des ressources de stockage

De manière similaire aux ressources de calcul, DIRAC fournit une interface standard pour accèder à plusieurs types de systèmes de stockage (*Storage Element*, SE). Cette interface est réalisée à travers un système de *plugins*, chacun dédié à un système de stockage particulier. Les systèmes actuellement supportés sont ceux les plus utilisés dans l'environnement grille (dCache, DPM, STORM, XROOTD, CASTOR, EOS, etc.) ainsi que ceux utilisés dans les environnements cloud (dépôts d'objets, S3). Cette interface permet d'effectuer de manière uniforme des opérations sur les fichiers et les répertoires présents sur différents types de *Storage Element*.

2.8.2 Service de Catalogue

Dans un environnement distribué, il est nécessaire de disposer d'un service de catalogue de fichiers dans lequel les fichiers sont référencés de manière unique, indépendamment de leur localisation physique. Ce service de catalogue référence typiquement des dizaines de millions de fichiers et il est très fréquemment interrogé par le WMS ainsi que par le gestionnaire de *workflows*. Il doit donc être fiable, performant et permettre le passage à l'échelle.

Pour répondre à ce besoin, un service de catalogue de fichiers, nommé LFC (LHC File Catalog), fut développé dans le cadre du *middleware* gLite. Depuis, plusieurs services de catalogue ont vu le jour et remplacé le LFC, qui a été décommisionnée en 2015.

Dans ce contexte, DIRAC a également développé un service de catalogue, nommé DIRAC File Catalog (DFC). Il est à noter que le DFC comporte deux types de catalogues avec des fonctionnalités différentes.

Le premier est le catalogue de fichiers (avec des fonctionnalités analogues au LFC). Il permet de référencer les fichiers dans un espace de nommage, il supporte des métadonnées de bas-niveau (taille de fichiers, adresse physique, checksum, etc.) et les droits d'accès. Chaque fichier référencé dans le catalogue possède un nom unique (*Logical File Name* ou LFN) dans cet espace de nommage.

Le deuxième type est le catalogue de métadonnées, qui supporte des métadonnées de haut niveau, c'est à dire des métadonnées définies par l'utilisateur et qui sont propre au domaine d'application. Des exemples de métadonnées définies dans le cadre des simulation de CTA sont donnés dans la Section 3.4.3. Ces métadonnées de haut niveau permettent en particulier de caractériser les données et de les regrouper en *datasets*, définis comme la liste de fichiers qui résulte d'une requête sur les métadonnées de haut niveau. Les *datasets* sont les entités qu'on expose généralement aux utilisateurs et qui sont utilisés pour l'enchainement des différentes étapes dans un *workflow*.

2.9 La gestion asynchrone des requêtes

Dans les sections précédentes, nous avons vu que le fonctionnement de DIRAC comporte l'exécution de nombreuses requêtes. La récupération des requêtes terminées en échec est essentielle pour garantir le bon fonctionnement de DIRAC (communications internes entre les différents composants) et le maximum de taux de réussite des requêtes venant des utilisateurs et des tâches. Le *Request Management System* (RMS) de DIRAC est en charge de la récupération de ces requêtes et de leur exécution asynchrone.

Les types d'opération supportés par le RMS sont de différente nature. Une application habituelle concerne l'implémentation d'un mécanisme de *Failover* qui intervient lorsqu'une tâche à la fin de son exécution n'arrive pas à copier les données produites sur un *Storage Element*. Le mécanisme de *Failover* consiste à copier les données vers un autre *Storage Element* et à créer une requête de transfert depuis ce dernier vers le SE de destination. Cette requête est stockée dans une base de données et traitée de manière asynchrone par le RMS.

Un autre cas d'application du RMS consiste à effectuer de manière asynchrone des opérations massives sur les données qui seraient trop longues à traiter de manière interactive, e.g. l'effacement de *datasets* de taille conséquente, transfert de données, etc. (cf. Section 2.10.2).

Enfin, il est à noter que le RMS peut supporter des opérations tout à fait génériques, qui peuvent au besoin être étendues par les utilisateurs.

2.10 La gestion des workflows

Dans des nombreux domaines (physique, astrophysique, bio-informatique, etc.) le traitement des données comporte plusieurs étapes de calcul inter-dépendantes qui forment un *workflow*. Dans un contexte de traitement massif de données, chacune de ces étapes est généralement composée d'un grand nombre de tâches individuelles traitant des *datasets* de l'ordre des To-Po. L'efficacité globale de ces traitements repose entre autre sur la capacité à automatiser l'enchainement des différentes étapes d'un *workflow*.

Plusieurs gestionnaires de *workflow* (*Workflow Management System*) existent actuellement sur le marché précisément pour répondre à ce besoin. Il est à noter qu'on distingue ici les fonctionnalités de *Workflow management* et de *Workload management*. La première se réfère à la création des tâches d'un *workflow* sur la base d'une description formelle de celui-ci, alors que la deuxième se réfère à la soumission et au suivi des tâches sur une infrastructure distribuée. Dans DIRAC ces fonctionnalités sont réalisées par des systèmes séparés. En revanche dans la littérature, le terme *Workflow Management* est souvent utilisé pour se référer aux deux fonctionnalités à la fois.

Selon la référence [23], les gestionnaires de *workflow* peuvent être classés en fonction des propriétés suivantes : **1)** modèle d'exécution (ou structure) du *workflow* ; **2)** environnement d'exécution ; **3)** méthodes d'accès aux données. Dans cette classification, on distingue cinq modèles d'exécution appartenant à deux groupes principaux (cf. Figure 2.6) : **1)** le groupe *acyclic* (DAG) ; **2)** le groupe *cyclic* non-DAG. Dans le modèle *acyclic* 'séquentiel', les tâches sont exécutées l'une après l'autre, lorsque la première tâche est terminée, ses résultats sont utilisés en entrée par une deuxième tâche (tâches 'faiblement couplées'). Dans le modèle *acyclic* 'concurrent', deux tâches progressent simultanément et les résultats de la première

sont consommés en temps réel par la deuxième, par exemple via un flux de données. Le modèle *cyclic* 'itératif' comporte des structures dans lesquelles certaines tâches peuvent être répétées de manière itérative. Le modèle *cyclic* avec 'pilotage externe' a une sémantique similaire au modèle concurrent, mais les résultats de certaines tâches sont inspectés par un système externe qui décide d'éventuellement modifier le comportement du *workflow*. Enfin, dans le cas du modèle *cyclic* 'étroitement couplé', deux tâches progressent en parallèle en s'échangeant des données en temps réel ('tâches couplées').

Par rapport à cette classification, les workflows HEP s'appuient sur un modèle d'exécution *acyclic* séquentiel et comportent des tâches 'faiblement couplées' qui s'exécutent de manière indépendante et qui sont connectées par les données produites à la fin de l'exécution.

Issu de la communauté HEP, DIRAC a été conçu pour supporter ce dernier type de modèle (*acyclic* séquentiel). Néanmoins, dans la Section 2.10.1 nous verrons que DIRAC supporte également le modèle concurrent, non pas au niveau des tâches individuelles, mais au niveau d'un ensemble de tâches à travers le concept de 'transformation'. En effet, une transformation se compose d'un ensemble de tâches qui consomment les données produites par une autre transformation au fur et à mesure que celles-ci sont produites. En outre, le comportement d'un *workflow* peut être modifié dynamiquement sur la base de la caractérisation des données produites par les tâches (système 'guidé par les données').

En ce qui concerne l'environnement d'exécution et les méthodes d'accès aux données, DIRAC présente des fonctionnalités avancées par rapport à la plupart des gestionnaires de *workflow*. En effet, DIRAC permet d'exécuter les tâches d'un même *workflow* dans des environnements différents et sur des ressources distribuées. Parmi les différentes méthodes d'accès aux données, DIRAC supporte aussi bien un accès au disque local de la ressource de calcul, qu'un accès à des systèmes de stockage distants à travers des protocoles de transfert (e.g. GridFTP, XRootD). Enfin, le système de *workflow* de DIRAC a été conçu pour gérer un grand nombre de tâches traitant une grande quantité de données, ce qui est primordial pour les applications HEP.

2.10.1 La gestion des workflows dans DIRAC

Dans DIRAC un *workflow* est conçu comme une série de transformations opérées sur les données, tel qu'illustré schématiquement sur la Figure 2.7. Chaque transformation est composée d'un ensemble de tâches opérant sur un *dataset* donné. La création automatique et la soumission de ces tâches au WMS sont assurées par le *Transformation System*. Ensuite, le *Production System* est en charge de l'enchainement de plusieurs transformations pour l'exécution d'un *workflow* complet. Ces deux systèmes fournissent ensemble les fonctionnalités nécessaires à la gestion des *workflows* dans DIRAC.

Dans les collaborations en physique, les systèmes de gestion de *workflows* sont utilisés par une petite équipe en charge des productions à grande échelle pour le compte de la collaboration. Typiquement, le gestionnaire de production définit un certain nombre de *workflows* standards utilisant les applications et les configurations officielles fixées par la collaboration. En parallèle de ces productions, de nombreux utilisateurs effectuent des analyses de haut niveau, impliquant moins de tâches et des plus petits volumes de données, en s'adressant directement au WMS.

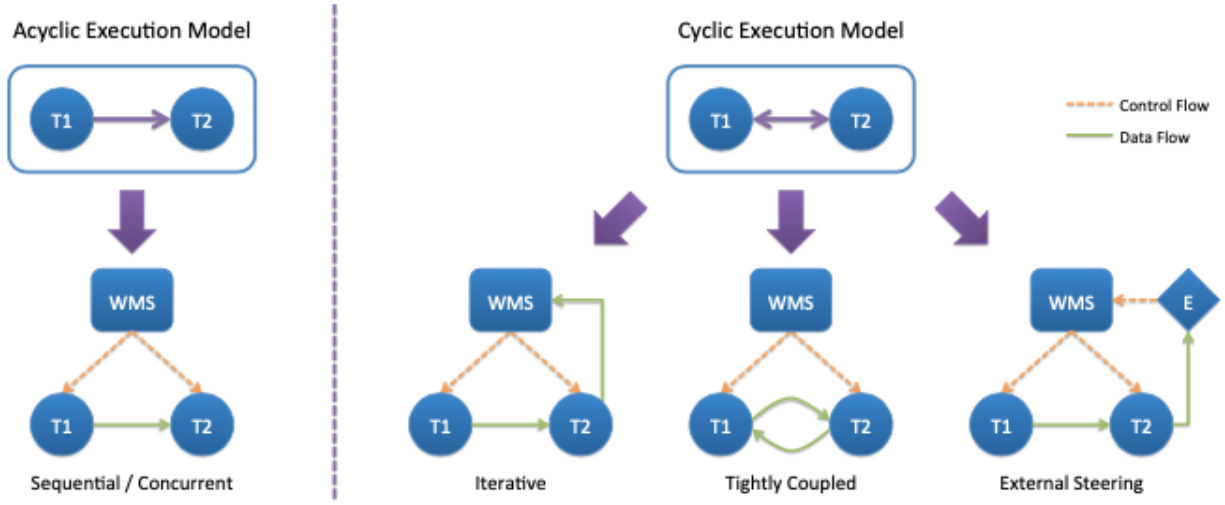

FIGURE 2.6 – Différents modèles d'exécution de *workflows* subdivisés en deux groupes : à gauche le groupe des modèles *acyclic* ; à droite le groupe des modèles *cyclic*. **Modèle séquentiel** : le *Workflow Management System* (WMS) démarre la tâche T1, lorsque celle-ci est terminée, T2 démarre et traite les données produites par T1. **Modèle concurrent** : T1 et T2 s'exécutent en même temps, T1 produit des données qui sont consommées en temps réel par T2. **Modèle itératif** : le WMS démarre T1, lorsque celle-ci est terminée, T2 est démarrée. En fonction du résultat de T2, une autre itération de cette séquence sera éventuellement exécutée et ainsi de suite. **Modèle étroitement couplé** : T1 envoie des résultats partiels à T2 et vice et versa. **Pilotage externe** : modèle concurrent dans lequel les résultats de T2 sont inspectés par un utilisateur ou un système externe qui éventuellement décide de modifier le comportement du *workflow*.

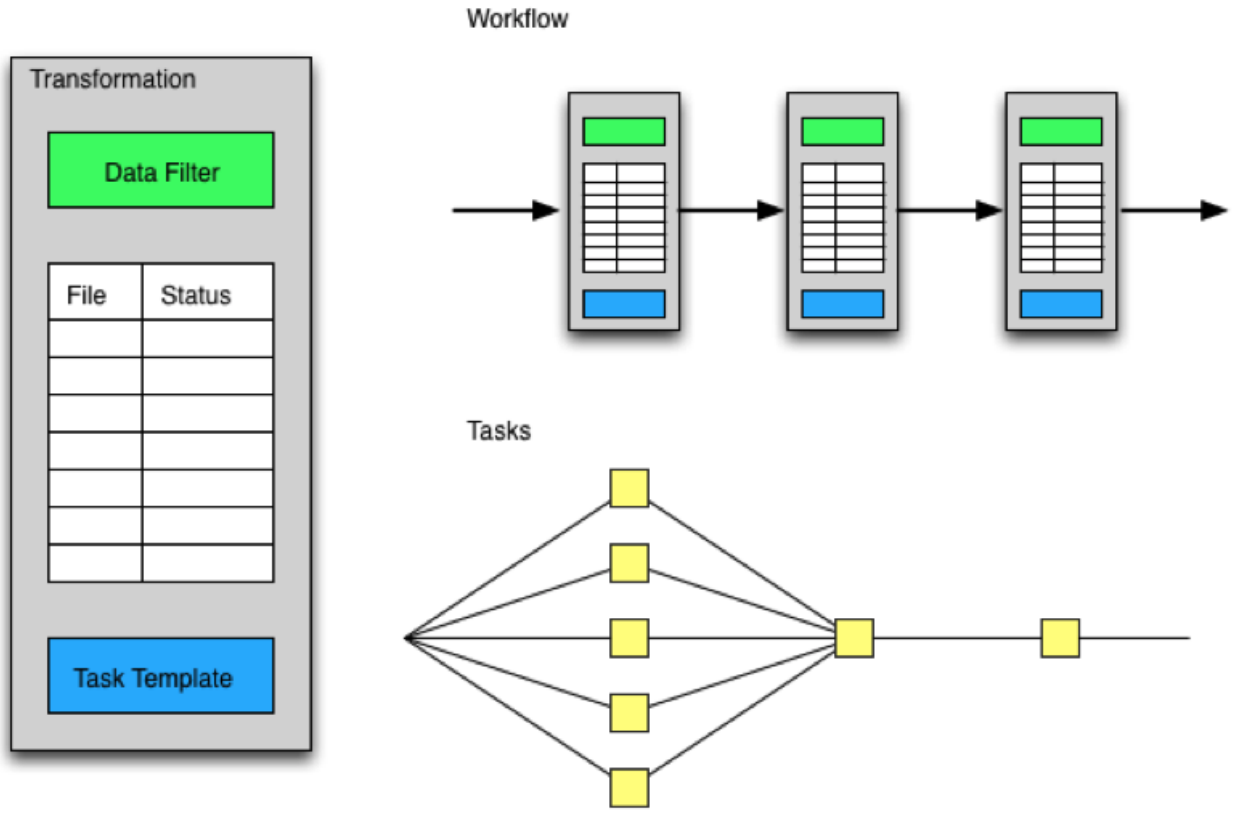

FIGURE 2.7 – En haut à droite, vue schématique d'un *workflow* constitué d'un enchainement de plusieurs transformations, chaque transformation étant composée par un ensemble de tâches (*Tasks* en bas à droite). A gauche, vue schématique d'une transformation définie par un modèle de tâche (*Task Template*) et un filtre sur les données (*Data Filter*).

2.10.2 Transformation System

En termes généraux, une transformation consiste en un ensemble de tâches 'similaires' traitant un *dataset* donné. Le *Transformation System* (TS) est en charge de créer les tâches associées à chaque transformation et de les soumettre au WMS. Le TS est donc un client du WMS, qui à son tour se charge de la soumission sur l'infrastructure de calcul. Plus précisément, une transformation dans DIRAC est définie par un modèle de tâches (*Task Template*) et par un filtre sur les données (*Data Filter*) à traiter.

Le modèle de tâches fournit une description générale des tâches en termes d'applications à exécuter, paramètres des applications, spécifications CPU, etc. Les tâches définies par ce modèle sont toutes identiques sauf pour un paramètre variable, qui peut être de type 'fichier' ou bien un paramètre de l'application exécutée. Grâce à ce modèle, il est donc possible de définir des transformations traitant ou non des données. Par exemple des transformations de type Monte-Carlo n'ont pas de données d'entrée, mais les tâches diffèrent par un paramètre de l'application. Au contraire, les transformations de traitement sont constituées de tâches identiques traitant chacune un ou plusieurs fichiers d'entrée différents.

Le filtre sur les données permet de définir le *dataset* à traiter par la transformation. Ce filtre est exprimé sous la forme d'une requête sur les métadonnées de haut niveau adressée au service de catalogue (DFC ou catalogue externe).

Sur la base du modèle de tâches et du filtre, le *Transformation System* crée les tâches correspondantes et leur associe les fichiers d'entrée. Cette association peut être réalisée selon différents critères (Plugins) également définis par l'utilisateur. Le TS supporte un certain nombre de Plugins 'standards' qui peuvent être facilement étendus par les utilisateurs. Le Plugin par défaut associe des fichiers à une tâche selon leur localisation physique. Le nombre de fichiers par tâche est également spécifié par l'utilisateur.

Une des caractéristiques fondamentales du TS est que l'exécution des transformations et in fine des *workflows*, est complètement 'guidée par les données' (*data-driven*). Cette caractéristique distingue DIRAC d'autres systèmes de gestion de *workflow* qui se reposent plutôt sur le statut des tâches. Le résultat de l'application du filtre sur les données étant dynamique, dès que des nouveaux fichiers correspondant au filtre sont enregistrés dans le catalogue, le TS crée automatiquement les tâches associées.

Enfin, le TS permet également de définir des règles pour distribuer les tâches dans les différents sites. La règle par défaut consiste à déterminer le site en fonction de la localisation physique des fichiers d'entrée. Cette règle permet d'optimiser l'accès aux données en évitant des transferts sur le réseau. Une autre règle très utilisée est le *ByJobTypeMapping*, qui permet de sélectionner les sites en fonction de critères de 'voisinage' (e.g. sites ayant une bonne connectivité réseau) définis par l'utilisateur et ce pour différents types de tâches. Cette règle autorise une grande souplesse dans la distribution des tâches et permet en particulier d'implémenter n'importe quel modèle de calcul.

2.10.2.1 Architecture logicielle

La Figure 2.8 illustre schématiquement comment ces fonctionnalités sont implémentées dans le TS. Il est à noter en outre que le TS est conçu comme un système général pour l'automatisation de tâches de différents types. En particulier, dans le TS les tâches peuvent êtres des tâches de calcul ou bien des requêtes sur les données qui sont envoyées respectivement

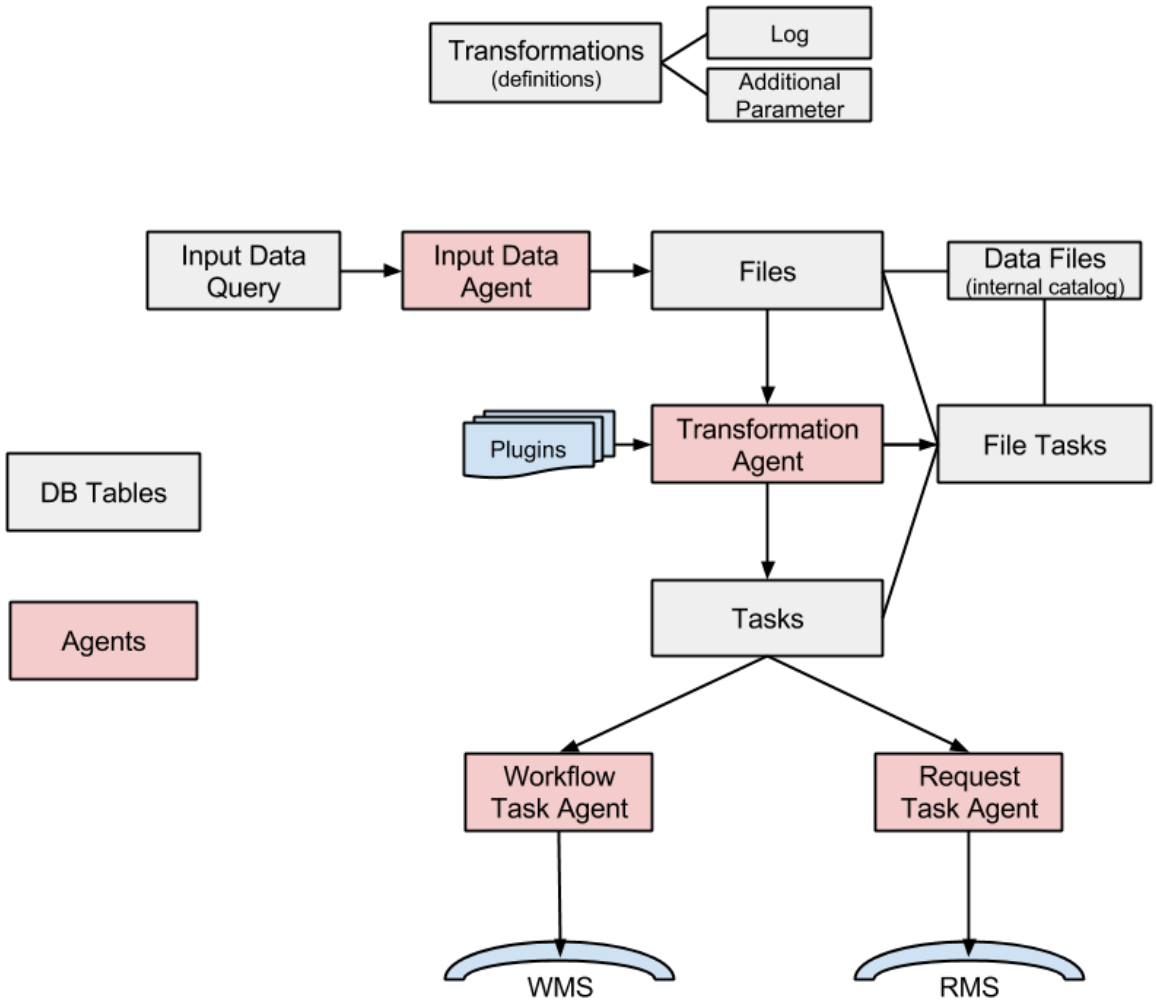

FIGURE 2.8 – Les transformations définies par l'utilisateur sont stockées dans une table dédiée (*Transformations*). Les filtres associés sont également sauvegardés dans une autre table (*InputDataQuery*). L'*InputData Agent* exécute les requêtes associées aux filtres et insère les fichiers sélectionnés dans la table *Files*. Le *Transformation Agent* vérifie l'état de cette dernière table afin de créer les tâches associées qui seront stockées dans la table *Tasks*. Ces tâches sont créées selon les Plugins associés (règles pour le regroupement de fichiers d'entrée et pour le choix du site d'exécution). Si les tâches créées sont des tâches de calcul, elles sont traitées par le *WorkflowTask Agent* qui les soumet au WMS, alors que si les tâches sont des requêtes, le *RequestTask Agent* les soumet au RMS.

au WMS ou au RMS. De cette manière, le TS supporte également des activités de gestion de données, comme la réplication massive ou l'effacement d'un *dataset*. Grâce à cette généralisation, il est possible de construire des *workflows* dans lesquels on combine des *workflows* de traitement et des opérations de gestion de données. Un cas d'application typique est celui d'un *workflow* composé de plusieurs transformations de traitement produisant un *dataset* de haut niveau destiné aux utilisateurs finaux qui est automatiquement répliqué sur plusieurs sites pour en faciliter l'accès et assurer la redondance.

2.10.2.2 Evolution de l'architecture logicielle

Ma contribution au *Transformation System* a consisté à introduire une nouvelle méthode pour la sélection des fichiers et leur association aux tâches. Les détails de ce développement sont décrits dans [24]. En effet, l'architecture décrite sur la Figure 2.8 présente quelques limitations en termes de performances. Une des limitations principale est due au fait que l'*InputData Agent* effectue en permanence des requêtes au catalogue de fichiers qui sont potentiellement assez longues. L'association des fichiers aux tâches et donc leur création en résulte particulièrement ralentie.

Dans l'architecture que j'ai proposée, l'association des fichiers aux transformations se fait en temps réel au moment de l'enregistrement des fichiers dans le catalogue. Pour implémenter ce comportement, j'ai ajouté au TS une interface de type 'catalogue' (*TS Catalog*), i.e. une API pour l'enregistrement, l'effacement des fichiers, etc. DIRAC peut supporter en effet plusieurs catalogues à la fois (internes ou externes à DIRAC), du moment où ils présentent une interface standard. J'ai donc instrumenté le *TS Catalog* pour que l'enregistrement d'un fichier corresponde à son association aux transformations ayant le 'bon' filtre. De cette manière, lorsqu'un fichier est enregistré dans le catalogue de fichiers (e.g. le DFC), ses métadonnées sont évaluées par rapport aux filtres des transformations. Si le fichier correspond à un ou plusieurs filtres, il sera immédiatement associé aux transformations. Grâce à ce changement d'architecture, j'ai renforcé le paradigme 'guidé par les données' et réduit considérablement le temps de création des tâches.

2.10.3 Production System

Les *workflows* utilisés dans les traitement de données scientifiques sont généralement composés de plusieurs transformations. Afin d'automatiser complètement l'exécution d'un *workflow*, il est donc nécessaire de pouvoir créer et monitorer toutes les transformations qui le composent.

Pour répondre à ce besoin, plusieurs communautés d'utilisateurs de DIRAC ont développé une couche logicielle plus ou moins complexe au dessus du *Transformation System*. Le système plus complexe est celui de LHCb, décrit dans [25]. Or, tous ces systèmes ont été conçus avec des spécificités propres à une communauté particulière. Dans le système de LHCb par exemple, les *workflows* sont créés à partir de modèles spécifiques à l'expérience, définis au préalable et stockés dans une base de données.

En revanche, d'autres communautés d'utilisateurs comme CTA ne disposaient d'aucun système de haut niveau pour gérer les transformations. C'est pourquoi, j'ai choisi de développer un nouveau système intégré dans DIRAC, appelé *Production System* (PS), assez

générique pour être utilisable par plusieurs communautés. Un des concepts de base de ce système consiste à créer des transformations connectées entre elles sur la base d'une caractérisation et une selection dynamique des données.

2.10.3.1 Concepts

La fonctionnalité principale du *Production System* consiste, à partir d'une description formelle d'un *workflow*, à créer et monitorer les différentes transformations qui le composent. Dans ce système, les transformations sont connectées entre elles sur la base de la caractérisation des données. Dans la suite j'utiliserai indistinctement les termes '*workflow*' et 'production'.

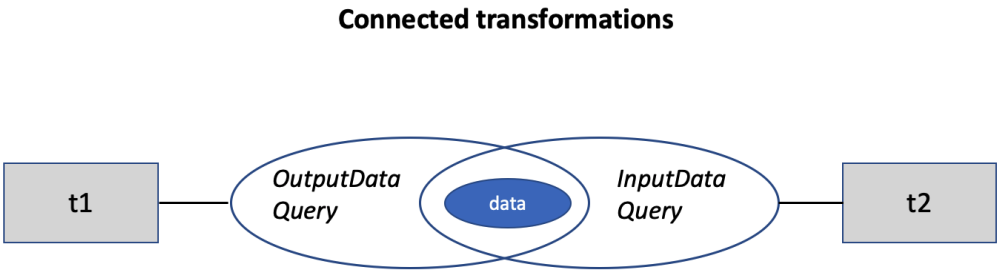

FIGURE 2.9 – Exemple de transformations connectées par les données. Des requêtes sur les métadonnées caractérisent les données en entrée et en sortie des transformations (*Input/OutputDataQuery*). Les transformations $t1$ et $t2$ sont connectées car il y a une intersection logique entre l'*OutputDataQuery* de $t1$ et l'*InputDataQuery* de $t2$. En particulier, les données produites par $t1$ (ou au moins une partie) sont utilisées en entrée de $t2$.

Une production est essentiellement définie comme un ensemble de transformations avec leurs associations. Plus précisément, une production est définie par un ensemble de *production steps*. Chaque *production step* contient la description d'une transformation ainsi que l'éventuelle spécification d'une 'transformation parente'. Deux transformations sont considérées comme étant connectées si elles ont une relation 'parent-enfant'. On appelle *Input/OutputDataQuery* les requêtes sur les métadonnées qui sélectionnent les *datasets* d'entrée et de sortie des transformations (l'*InputDataQuery* n'étant rien d'autre que le filtre sur les données associé à une transformation défini dans la Section 2.10.2).

Lorsqu'il y a une intersection logique entre l'*OutputDataQuery* d'une transformation et l'*InputDataQuery* d'une autre transformation, la première peut être déclarée comme 'parente' de la deuxième. En pratique cela signifie qu'au moins une partie des données produites par la première transformation sont utilisées en entrée par la deuxième transformation, comme dans l'exemple de la Figure 2.9. Une transformation peut avoir plusieurs 'transformations parentes' (ou aucune) et une ou plusieurs 'transformations enfants'. De cette manière, le *Production System* supporte plusieurs topologies de *workflow*, comme illustré sur la Figure 2.10. En combinant ces topologies, des *workflows* plus complexes peuvent être également exécutés.

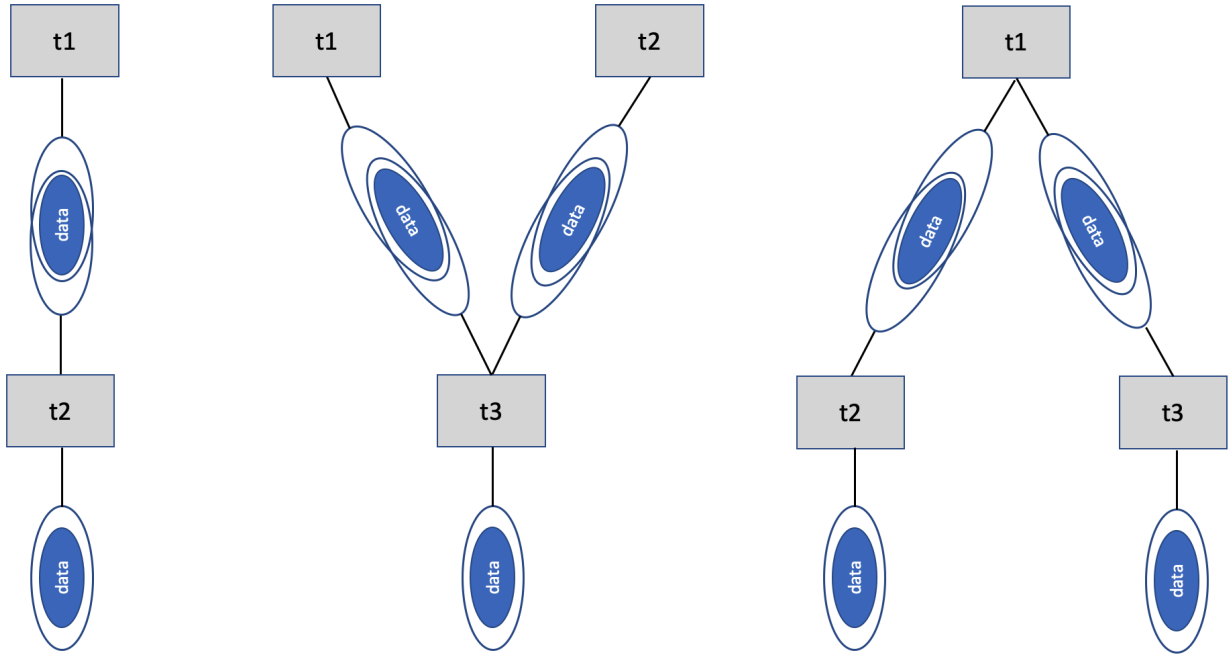

FIGURE 2.10 – Exemples des topologies de *workflow* supportées par le *Production System* de DIRAC. Les connexions entre les différentes transformations sont entièrement caractérisées par les données.

2.10.3.2 Architecture

Les composants principaux du *Production System* ainsi que leurs interactions avec les autres systèmes de DIRAC (TS et DFC) sont représentés schématiquement sur la Figure 2.11. De manière générale, j'ai conçu le *Production System* selon l'architecture standard d'un Système de DIRAC. En particulier il est composé de :

- La base de données *Production DB*
- Le service *Production Manager*
- L'API *Production Client*

La *Production DB* conserve de manière permanente les descriptions des productions ainsi que les relations (e.g. 'parent-enfant') entre les transformations. Le *Production Client* y accède au travers du service *Production Manager*. À ces composants de base, s'ajoutent 3 utilitaires ayant des fonctions spécifiques et qui gèrent les interactions avec le *Transformation System* et le DFC (ou un catalogue externe) :

- *ProdValidator* : vérifie la validité d'une production lors de sa création (interaction avec le DFC)
- *ProdTransManager* : gère les interactions avec le *Transformation System*
- *StateMachine* : assure la validité des transitions entre les différents statuts d'une production

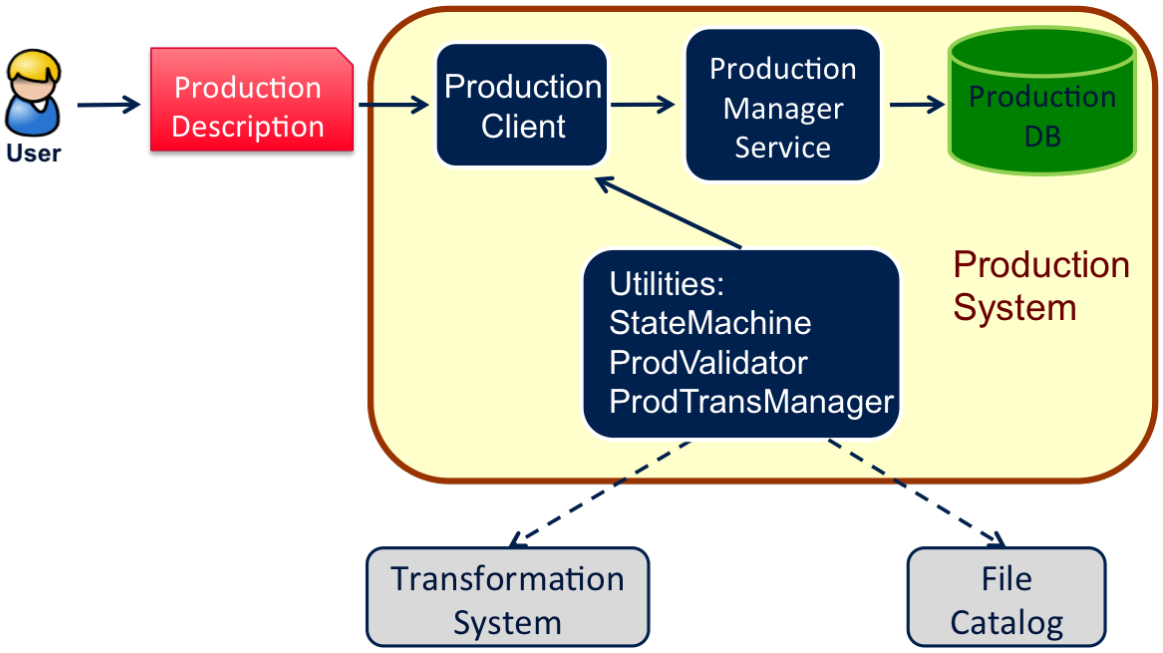

FIGURE 2.11 – Schéma du fonctionnement interne du *Production System*. **1)** L'utilisateur fournit une description de la production à travers le *Production Client*. **2)** Le *ProdValidator* vérifie que la production est valide en contrôlant que les différentes transformations sont bien connectées par leurs *Input/OutputDataQueries*. Une fois validée, la production est insérée dans la *Production DB*. **3)** Le *ProdTransManager* crée les différentes transformations associées à la production.

Plus de détails sur l'architecture du *Production System* sont donnés dans cette contribution [26].

Chapitre 3

Evaluation de DIRAC pour CTA

Dans la Section 1.4, j'ai exposé les rôles du gestionnaire des tâches (`CWMS`) et du système d'archivage. Dans ce chapitre, je vais présenter le prototype de gestionnaire de tâches que j'ai développé pour CTA en utilisant le framework DIRAC.

En 2011, j'ai initié ce projet, nommé `CTA-DIRAC`, avec deux objectifs principaux. Le premier consiste à évaluer les fonctionnalités de DIRAC vis-à-vis des spécifications du `CWMS` de CTA. L'application principalement visée est la gestion du traitement de données de bas niveau (du niveau DL0 au niveau DL3) et des simulations. J'ai à cette fin déployé une instance de DIRAC dédiée à CTA (`CTA-DIRAC`), permettant d'accèder aux ressources de la grille EGI et servant de prototype au futur `CWMS`.

Le deuxième objectif du projet consiste à exploiter ce prototype pour exécuter sur la grille les campagnes de simulation Monte-Carlo pendant les phases de préparation et de construction de CTA. L'idée est d'offrir un service au consortium et en même temps de profiter de ce cas d'utilisation concret pour peaufiner le prototype du `CWMS`. Le LUPM a donc pris la responsabilité de ces productions, que j'assure en coordination avec le groupe de travail *Analysis and Simulation Working Group* de CTA.

Grâce à l'exploitation du prototype dans ce contexte de simulations massives, j'ai pu non seulement effectuer une évaluation approfondie de DIRAC, mais également comprendre son fonctionnement interne et développer des couches logicielles supplémentaires pour les besoins de CTA. L'architecture de DIRAC permet en effet d'ajouter simplement une extension propre à chaque communauté d'utilisateurs. J'ai donc développé une extension propre à CTA, consistant en un jeu d'interfaces pour faciliter la configuration des tâches, des transformations et des productions de CTA. Ayant également identifié la nécessité d'automatiser au maximum l'exécution des *workflows*, j'ai développé le *Production System*, présenté dans la Section 2.10.3, qui est aujourd'hui intégré dans la release officielle de DIRAC.

Les travaux autour du projet `CTA-DIRAC` ont fait l'objet de plusieurs présentations et *proceedings*, voir [27] [28] [29] [30] [26].

3.1 Ressources grille pour CTA

Pendant la phase de préparation de CTA (2010-2017), les simulations Monte-Carlo ont nécessité des ressources de calcul et de stockage conséquentes, difficiles à obtenir auprès d'un

seul centre de calcul. C'est pourquoi en 2008, l'Organisation Virtuelle CTA (VO CTA) a été créée pour que les membres du consortium puissent accéder aux ressources de la grille EGI. Actuellement, environ une vingtaine de sites distribués dans sept pays en Europe supportent la VO CTA dont je suis l'administrateur.

Les ressources de calcul pour CTA sont fournies par les différents sites de grille sur une base volontariste, alors qu'à terme les centres seront liés par accords formels. Dans ce contexte, le consortium m'a confié la coordination de l'exploitation des ressources de la grille pour CTA, baptisée CTACG (*CTA Computing Grid*). Afin de garantir que le consortium dispose des ressources nécessaires pour réaliser son programme de simulation, j'ai instauré une procédure pour l'estimation et pour la demande de ressources.

Le groupe de travail *Analysis and Simulation Working Group* définit régulièrement, en lien avec plusieurs autres groupes (groupes travaillant sur les prototypes de caméras, groupes de physique, etc.) un programme de simulations à réaliser. Sur la base de ce programme, on estime annuellement les ressources CPU et de stockage nécessaires. Ensuite, nous présentons notre demande de ressources au cours d'une réunion annuelle avec les sites CTACG. Enfin, les sites nous communiquent le quota de ressources qu'ils s'engagent à allouer à CTA pour l'année à venir. Les ressources rassemblées sont loin d'être négligeables, comme le montrent les graphes d'utilisation de la grille sur la Figure 3.7. Environ 10k coeurs sont disponibles en simultané lors des phases de production intensives. Enfin, six sites fournissent également du stockage disque, pour un total de 5 Po et trois d'entre eux du stockage sur bandes magnétiques (environ 2 Po).

3.2 Modèle de calcul

Le modèle de calcul utilisé pour les simulations Monte-Carlo dans les phases de préparation et construction de CTA est un modèle simplifié par rapport à celui prévu à terme pour les opérations (cf. Section 1.4.3).

Dans ce modèle, on distingue deux activités principales coordonnées par une petite équipe production (constituée actuellement par J. Bregeon et moi même) : **1)** la production Monte-Carlo ; **2)** l'analyse de données Monte-Carlo. À cela s'ajoute l'activité des utilisateurs qui utilisent la grille pour des simulations ou des analyses spécifiques.

Les tâches des productions Monte-Carlo sont exécutées sur l'ensemble des sites disponibles, alors que les données produites par ces tâches sont transférées directement sur les six sites fournissant du stockage.

Au total, nous produisons une seule copie des différents *datasets*, distribuée sur les six sites. Aucune réplication de ces *datasets* est effectuée, du fait des ressources de stockage limitées. L'utilisation des bandes magnétiques est également différente par rapport au modèle final. Actuellement, nous utilisons les bandes magnétiques uniquement pour archiver les *datasets* les plus anciens avant leur effacement définitif.

Concernant les tâches d'analyse, dans un premier temps nous avons exigé qu'elles soient exécutées au plus proche des données, donc sur les six sites où sont localisées les données à traiter. Ce modèle d'accès local est assez performant car il minimise les transferts sur le réseau. En contrepartie, les ressources CPUs disponibles pour le traitement en résultent limitées. Dans un deuxième temps, nous avons adopté un modèle plus flexible, dans lequel des

sites supplémentaires sont autorisés à exécuter des tâches d'analyse (modèle d'accès distant).

La distribution des tâches d'analyse dans les différents sites est obtenue à l'aide d'un Plugin spécifique du *Transformation System* (le Plugin *ByJobTypeMapping* décrit dans la Section 2.10.2).

Enfin, les tâches des utilisateurs sont exécutées en parallèle aux activités principales de production et d'analyse. Ayant moins de contrôle sur les *workflows* des utilisateurs, pour cette catégorie de tâches nous avons préféré garder un modèle d'accès local.

Ce modèle simple et flexible s'est avéré suffisant pour supporter les différentes campagnes de simulation effectuées sur la grille depuis 2013. En particulier, bien que l'absence d'un mécanisme de réplication de données pose des problèmes de sécurité, cet aspect n'est pas critique s'agissant de données simulées. D'ailleurs, suite aux inondations du centre de calcul CNAF en Italie en 2017, nous avons subi une perte de plusieurs centaines de To que nous avons pu aisément reproduire. En outre, l'introduction d'un modèle d'accès distant pour les tâches d'analyse a permis de réduire significativement les délais de traitement.

Enfin, bien que ce modèle soit relativement simple par rapport au modèle final, l'utilisation d'un plus grand nombre de centres, ayant des niveaux de disponibilité très différents, pose des difficultés opérationnelles dans le suivi des productions. Nous nous attendons donc à ce que le coût en ressources humaines pour le suivi des productions soit réduit dans le modèle final avec seulement 4 centres.

3.3 Le prototype CTA-DIRAC

3.3.1 L'infrastructure matériel

Le prototype `CTA-DIRAC` a évolué depuis sa première installation en 2011 qui comportait seulement deux serveurs installés au centre de calcul de PIC à Barcelone. Actuellement, les différents composants de DIRAC sont déployés sur cinq serveurs dans deux centres de calcul principaux (CC-IN2P3 à Lyon et PIC à Barcelone), plus un serveur additionnel à DESY (Berlin) pour assurer la redondance de certains services.

Au CC-IN2P3, les composants sont déployés sur deux serveurs principaux comme suit :

- serveur-01 : *Data Management System*
- serveur-02 : *Transformation*, *Production*, *Request Management Systems*

Un troisième serveur (serveur-03) est dédié au portail web de DIRAC. Au centre de calcul de PIC, deux serveurs supplémentaires sont dédiés aux composants suivants :

- serveur-04 : *Workload Management System*, *Configuration System - Master*
- serveur-05 : *Framework*, *Accounting*, VMDIRAC

Afin d'assurer le maximum de redondance du *Configuration System*, sur chacun de ces serveurs nous avons également déployé une instance *Slave* de ce dernier (cf. Section 2.5). Les bases des données correspondantes aux différents systèmes sont également hébergées dans les deux centres de calcul sur des infrastructures de haute disponibilité. Enfin, sur le serveur à DESY nous avons dupliqué certains services critiques, comme le DFC, pour ajouter de la redondance.

L'infrastructure décrite ci-dessus permet de supporter les activités de simulations actuelles, i.e. la charge induite par plus de 10k tâches concurrentes effectuant chacune plusieurs requêtes aux différents services DIRAC. À terme, en vue des opérations de CTA, il sera aisé de redimensionner cette infrastructure en termes de capacité et nombre de serveurs en fonction de la charge.

En revanche, le prototype `CTA-DIRAC` dans son état actuel ne garantit pas une haute disponibilité des services. En cas de panne ou d'indisponibilité d'un serveur, les services DIRAC correspondants ne seront pas disponibles. Pour atteindre un niveau de production, il faudrait assurer la redondance des différents composants sur plusieurs serveurs géographiquement distribués. Au niveau du prototype, nous avons toutefois porté une attention particulière à la sécurité des bases de données, qui conservent l'état des différents systèmes. Une sauvegarde journalière a donc été mise en place par les administrateurs des centres calcul, permettant ainsi de redéployer entièrement l'instance `CTA-DIRAC` en cas de panne sévère.

3.4 Les simulations Monte-Carlo

Pendant la phase de préparation de CTA (2010-2017), le consortium a effectué plusieurs campagnes de simulations afin d'étudier les performances attendues et optimiser le réseau. La première campagne de simulation massive, nommée PROD1 et menée en partie sur la grille de calcul en 2011/2012, a permis de produire les premières fonctions de réponses de référence de CTA, bien que réalisée avec une description approximative des télescopes et sur des sites géographiques génériques. En 2013/2014, la seconde campagne massive, nommée PROD2, avait pour objectif de comparer les performances d'un réseau donné de télescopes en fonction du site d'implantation géographique. À l'époque plus de six sites étaient proposés pour accueillir l'un des deux réseaux de CTA. Après le choix des sites, la troisième production PROD3 a été lancée en 2015/2016 pour optimiser la géométrie des réseaux, en particulier l'espacement entre les télescopes. En 2017/2018, la PROD3(b) a permis de vérifier les derniers détails de la géométrie, et de produire des fonctions de réponses avec des erreurs statistiques réduites pour une partie significative de l'espace des paramètres (angles zénithaux et azimutaux, bruit de fond de ciel, etc.). En 2019 la PROD4 a été dédiée à une étude comparative des différents modèles proposés pour les télescopes de petite taille. Après cette phase d'optimisation, la dernière campagne a été lancée en 2020 (PROD5) afin de produire de nouvelles fonctions de réponses de référence et calculer la sensibilité attendue avec une description mise à jour des télescopes.

Depuis 2012, j'assure ces productions pour le consortium, en exploitant le prototype `CTA-DIRAC` et les ressources de la grille, selon le modèle de calcul présenté dans la Section 3.2. Les études Monte-Carlo mentionnées ci-dessus s'appuient sur la production d'une grande statistiques d'événements en utilisant les pipelines de simulation et de traitement décrits dans la Section 1.4.1. Dans cette section, je vais détailler quelques exemples des *workflows* que nous avons utilisé pour exécuter ces pipelines à grande échelle, en expliquant le rôle des différents composants de DIRAC dans la gestion de ces *workflows*. Enfin, je vais présenter le résultats des études Monte-Carlo pour deux campagnes majeures de simulations, i.e. celles dédiées au choix des sites et de la géométrie du réseau.

3.4.1 Workflows et logiciels de simulation et de traitement

Le *workflow* utilisé dans les campagnes de simulation mentionné ci-dessus consiste dans les grandes lignes à exécuter le pipeline Monte-Carlo et à injecter les données de niveau DL0 ainsi générées dans le pipeline de traitement principal. Pour rappel ce dernier est composé de trois étapes produisant différents niveaux de données (DL1-DL3) : **1)** reconstruction des images (niveau DL1) ; **2)** reconstruction des événements (niveau DL2) ; **3)** sélection des événements (niveau DL3).

Actuellement seules les premières étapes de ce *workflow* sont exécutées sur la grille, i.e. jusqu'à la génération de données de niveau DL1. Les étapes successives nécessitent beaucoup moins de ressources et pour des questions pratiques sont exécutées actuellement sur les fermes de calcul des instituts impliqués dans l'analyse. Toutefois, à terme, l'objectif est d'exécuter le *workflow* complet sur les quatre centres de calcul prévus dans le modèle de calcul.

Aujourd'hui l'implémentation de ce *workflow* s'appuie sur les logiciels actuellement disponibles au niveau du consortium. Bien qu'à terme ces logiciels vont évoluer ou seront remplacés, l'enchainement logique des différentes étapes du *workflow* restera essentiellement inchangé.

Le pipeline Monte-Carlo repose sur le logiciel `CORSIKA` (*COsmic Ray SImulations for KAscade*) pour la simulation des cascades atmosphériques et sur le logiciel `sim_telarray` pour la simulation de la réponse optique et de l'électronique de la caméra.

`CORSIKA` est un logiciel de référence utilisé par de nombreuses expériences dans les domaines des rayons cosmiques, de l'astronomie gamma, radio et neutrino (e.g. Pierre Auger Observatory, CTA, IceCube, etc.). Ecrit en Fortran, sa première version date des années 90s. Du fait de la difficulté à maintenir ce code ancien, un projet collaboratif de ré-écriture complète de `CORSIKA` en C++ moderne a été lancé en 2018 par le Karlsruhe Institute for Technology (projet `CORSIKA 8` [31]). CTA bénéficiera donc à terme de cette nouvelle version.

Le programme `sim_telarray` hérite d'une architecture très ancienne et difficilement tenable sur la durée de CTA, mais c'est aujourd'hui le code le plus stable et robuste : il faudra pourtant nécessairement tôt ou tard le remplacer.

Pour le pipeline de traitement, il a été décidé au sein du consortium de le développer ex-novo, plutôt que d'adapter des logiciels existants. À cette fin, un framework en python (`ctapipe`), fournissant toutes les fonctionnalités nécessaires pour implémenter le pipeline de traitement, a été développé. Aujourd'hui, le pipeline de traitement est bien avancé mais toujours en cours de validation. C'est pourquoi trois logiciels adaptés des expériences existantes ont été utilisés ces dernières années pour l'analyse des simulations et la production des fonctions de réponse : `read_cta` adapté d'un logiciel de H.E.S.S., `EventDisplay` provenant de Veritas et `MARS` pour *MAGIC Analysis and Reconstruction Software*.

Une version simplifiée du *workflow* utilisé est représentée par un enchainement d'applications, où chaque application traite les données produites par l'application précédente. Sur la Figure 3.1, j'ai indiqué les logiciels actuellement utilisés, où par simplicité dans les étapes de traitement j'ai cité seulement le logiciel `EventDisplay`. En réalité, afin de vérifier la robustesse de l'analyse, nous avons souvent utilisé en parallèle les trois logiciels disponibles `read_cta`, `EventDisplay` et `MARS`.

Or, tel qu'illustré sur la Figure 3.2, le *workflow* effectivement utilisé est en peu plus complexe, car tous les logiciels de traitement s'appuient sur des méthodes d'apprentissage automatique pour l'estimation de la direction et de l'énergie de la cascade ainsi que pour la

réjection du bruit de fond. L'utilisation de ces méthodes implique qu'un *workflow* d'apprentissage soit exécuté au préalable ou partiellement en parallèle au *workflow* principal pour le calcul des modèles qui seront ensuite injectés dans l'étape de reconstruction (production de données de niveau DL2).

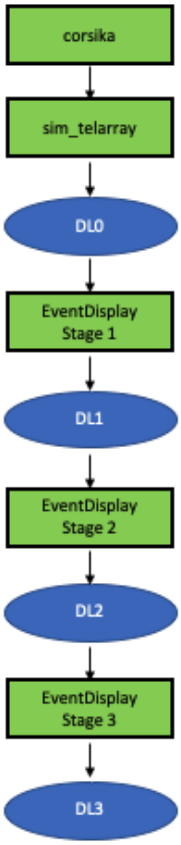

FIGURE 3.1 – Vue simplifiée du *workflow* utilisé dans les campagnes de simulation pour la génération des fonctions de réponse. Les logiciels utilisés dans les différentes étapes sont indiqués.

Actuellement on exécute sur la grille le *workflow* principal jusqu'au niveau DL1, alors que le *workflow* d'apprentissage ainsi que la sélection des événements pour produire le niveau DL3 sont effectués au niveau des instituts en charge de l'analyse. À terme, l'objectif est d'automatiser l'exécution du *workflow* complet jusqu'à la génération des fonctions de réponse pour la caractérisation des performances de l'instrument en incluant également la génération des modèles.

3.4.2 Utilisation de DIRAC pour la gestion des workflows

Le *workflow* décrit dans la section précédente peut être implémenté dans DIRAC à l'aide du *Transformation* et du *Production Systems* de plusieurs façons. Dans une tâche on peut exécuter une ou plusieurs applications, ce qui détermine également le nombre de transformations.

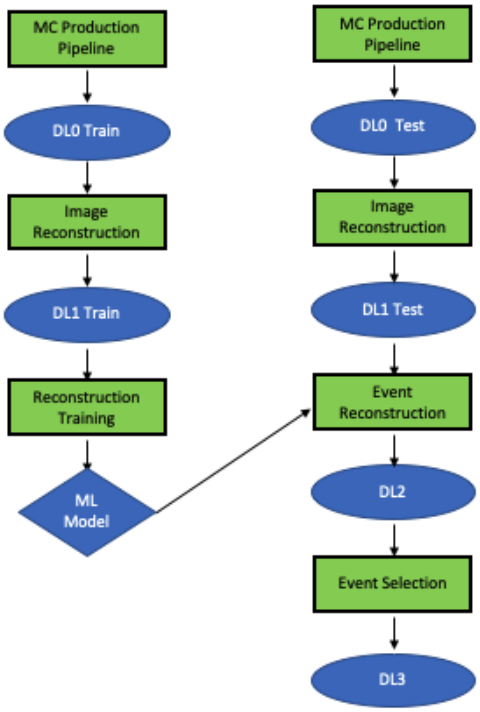

FIGURE 3.2 – *Workflow* pour la génération des fonctions de réponse utilisant des méthodes d'apprentissage automatique. À gauche, le *workflow* d'apprentissage produit des modèles, qui sont ensuite injectés dans l'étape de reconstruction du *workflow* principal.

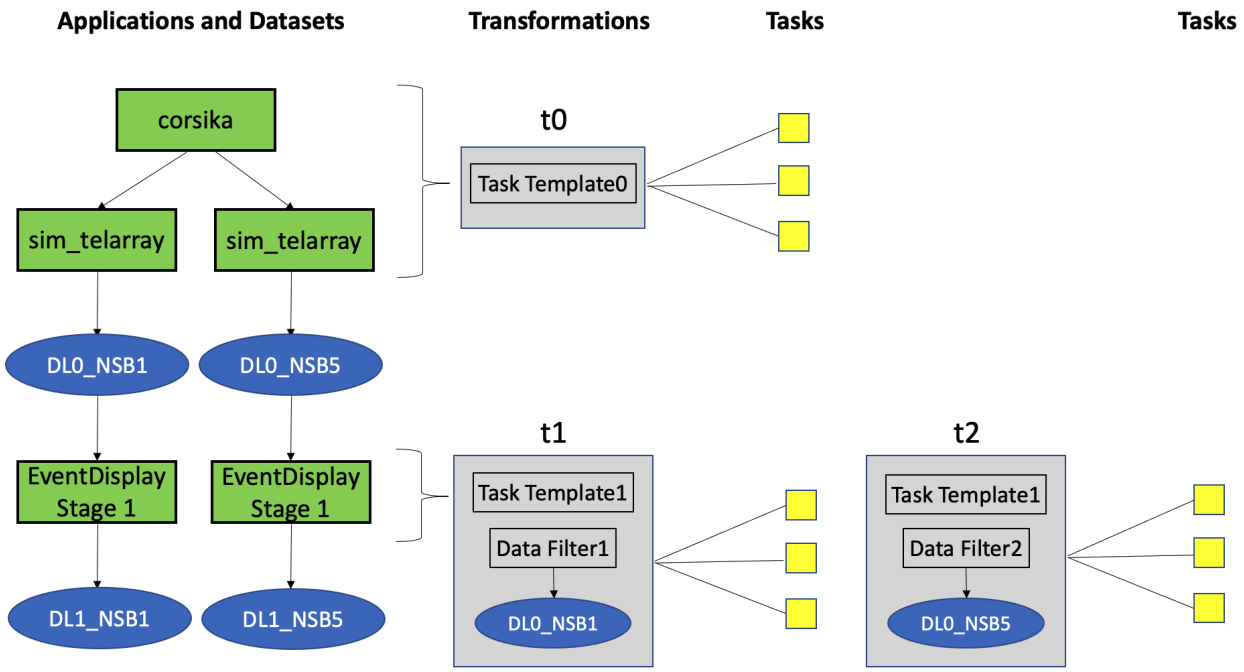

FIGURE 3.3 – Exemple de *workflow* utilisé pendant la PROD5 utilisant 3 transformations connectées.

Ces dernières peuvent ensuite être regroupées de plusieurs manières dans des productions, cf. 2.10.3.

À titre d'exemple, je vais décrire l'implémentation du *workflow* utilisé dans la dernière campagne (PROD5), qui présente une légère différence par rapport au *workflow* séquentiel de la Figure 3.1, du fait de la simulation en parallèle de deux niveaux de bruit de fond du ciel (*Night Sky Background* ou NSB). L'implémentation de ce *workflow* à l'aide du *Transformation System* est représentée sur la Figure 3.3. Tout d'abord, nous avons défini deux modèles de tâches. Le premier modèle ($TaskTemplate0$) définit des tâches exécutant en séquence `CORSIKA` et deux instances de `sim_telarray`, l'une pour $NSB = 1$ et l'autre pour $NSB = 5$. Deux fichiers de niveau DL1 sont produits par ce type de tâches correspondant aux deux valeurs de NSB égales à 1 et à 5. Le deuxième modèle ($TaskTemplate1$) définit des tâches exécutant l'application `EventDisplay_stage1`. Ces deux modèles sont ensuite utilisés pour définir, avec les filtres sur les données, les transformations correspondantes. Le premier modèle permet de définir une transformation sans données d'entrée qui exécute le pipeline de simulation ($t0$ sur la Figure 3.3). Les transformations de traitement ($t1$ et $t2$ sur la Figure 3.3) sont également définies par deux filtres sur les données (*InputDataQuery1* et *InputDataQuery2*), permettant de sélectionner respectivement les données produites pour $NSB = 1$ et $NSB = 5$. À partir de ces modèles et ces filtres, nous obtenons 3 transformations :

- $t0 = (TaskTemplate0)$
- $t1 = (TaskTemplate1,\ InputDataQuery1)$
- $t2 = (TaskTemplate1,\ InputDataQuery2)$

Il est à noter qu'afin de simuler des événements gamma ainsi que des événements de bruit de fond, ce *workflow* doit être exécuté pour 4 configurations différentes, correspondant aux 4 types de particules injectées dans la simulation : gamma source ponctuelle, gamma source diffuse, électron et proton. Il sera donc nécessaire de créer 3 transformations pour chacune de ces configurations pour un total de 12 transformations. En outre, il est également nécessaire de considérer deux directions de pointage différentes (Nord et Sud), ce qui double le nombre de transformations, pour un total de 24. On se rend compte ainsi que pour couvrir l'espace des paramètres (angles zénithaux et azimutaux, bruit de fond de ciel, site d'observation), le nombre de transformations nécessaires augmente considérablement.

C'est précisément ce type de cas d'application qui m'a poussé à développer le *Production System*. L'utilisation de ce système lors de la PROD5 a facilité largement la création et le monitorage des nombreuses transformations composant le *workflow*, tout en minimisant le risque d'erreurs lors de la définition de ce dernier. À titre d'exemple, nous avons regroupé les 24 transformations de la PROD5 en 8 productions, i.e. une pour chaque configuration (particule, direction de pointage), tel qu'illustré sur la Figure 3.4.

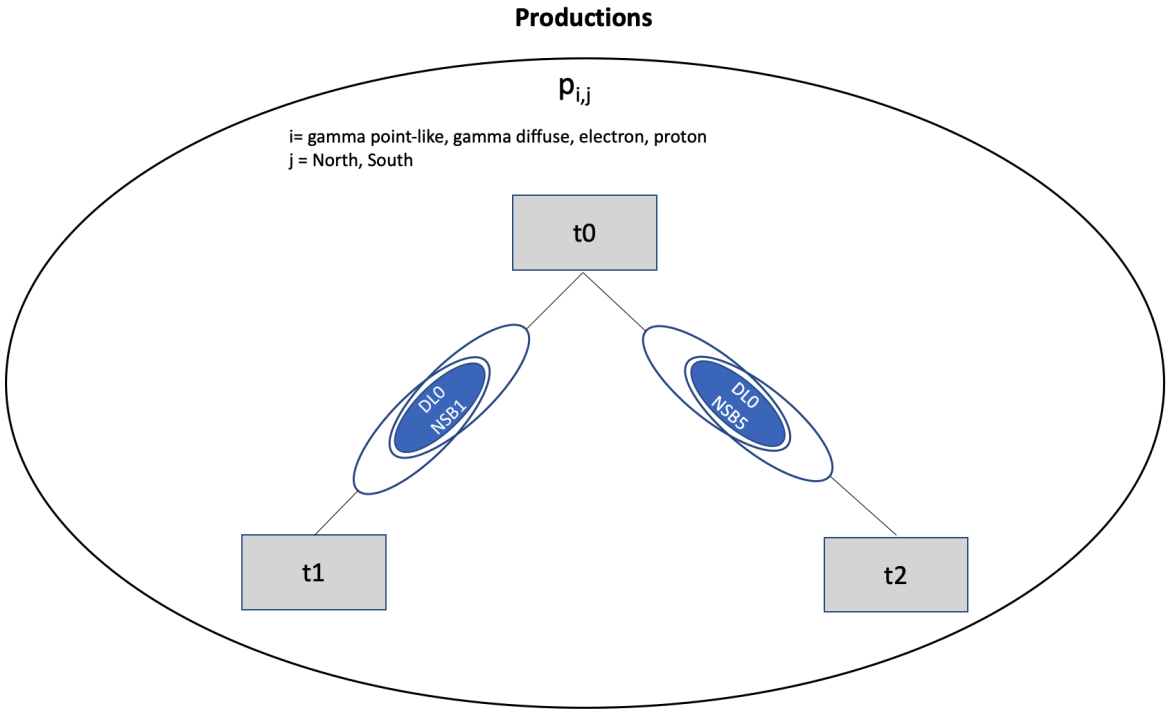

FIGURE 3.4 – Exemple de production utilisée pendant la campagne de simulation PROD5. Au total 8 productions ont été nécessaires pour simuler 4 types de particules incidentes et 2 directions de pointage.

3.4.3 Caractérisation par les métadonnées

Afin de caractériser les données de simulation, nous avons défini dans le catalogue DFC un ensemble de métadonnées :

```
zenithAngle, azimuthAngle, primaryParticle, site, arrayLayout, outputType,
dataLevel, airshowerSimProg, airshowerSimProgVersion, telSimProg,
telSimProgVersion, calibimgRecoProg, calibimgRecoProgVersion, MCCampaign,
nsb, runNumber, jobID
```

Des requêtes sur ces métadonnées sont utilisées pour définir les filtres des transformations (*InputDataQuery*, *OutputDataQuery*) qui composent les *workflows*. Les valeurs de ces métadonnées sont mises à jour directement par les tâches au moment de l'enregistrement des fichiers dans le DFC. Comme nous avons vu dans la Section 2.10.2, lors de cette opération d'enregistrement les fichiers sont également associés à toutes les transformations ayant le 'bon' filtre. À titre d'exemple, le filtre d'une transformation utilisée dans un *workflow* de la PROD5 a la forme suivante :

```
zenithAngle=20, arrayLayout=AdvancedBaseline, outputType=Data, dataLevel=0,
telSimProg=simtelarray, telSimProgVersion=v1r2, MCCampaign=PROD5, nsb=1,
azimuthAngle=0, site=Paranal, primaryParticle=gamma
```

Enfin, les différents filtres utilisés pour la PROD5 ont une forme similaire, mais avec des valeurs différentes pour certaines métadonnées, i.e. :

```
nsb=[1,5], azimuthAngle=[0,180], site=[Paranal,LaPalma],
primaryParticle=[gamma,gamma-diffuse,proton,electron]
```

3.4.4 Résultats des études Monte-Carlo

Afin de donner un aperçu de la finalité des productions Monte-Carlo, j'ai choisi de présenter les résultats obtenus pour deux campagnes de simulation majeures lors de la phase d'optimisation du réseau, notamment la PROD2 dédiée au choix des sites et la PROD3 dédiée au choix de la géométrie du réseau.

3.4.4.1 Choix des sites

L'objectif de la PROD2 (2014/2015) était de caractériser un certain nombre de sites géographiques, afin de déterminer les meilleurs sites pour l'implantation des réseaux de télescopes CTA dans les hémisphères Nord et Sud. En effet, trois paramètres principaux jouent un rôle important dans le développement des cascades atmosphériques et in fine des performances de CTA :

- La qualité de l'atmosphère au-dessus du site et en particulier sa transmission
- L'altitude du site et donc d'observation de la cascade. À basse altitude, la couche d'atmosphère traversée par les photons Cherenkov est plus importante ainsi que la probabilité que ces derniers diffusent avant d'arriver sur les télescopes
- Le champ géomagnétique orthogonal local, qui déforme le développement de la cascade électromagnétique. Plus le champ est intense et plus la cascade est dispersée, ce qui rend plus difficile sa reconstruction

TABLE 3.1 – Sites candidats pour l'implantation des télescopes de CTA considérés pour la campagne de simulation PROD2.

Nom du site	Lat., Long. [deg]	Altitude [m]	B_x [μT]	B_z [μT]
Aar (Namibie)	26.69 S 6.44 E	1640	10.9	-24.9
Armazones (Chili)	24.58 S 70.24 W	2100	21.4	-8.9
Leoncito@2640 m (Argentine)	31.72 S 69.27 W	2640	19.9	-12.6
Leoncito@1650 m (Argentine)	31.41 S 69.49 W	1650	19.9	-12.6
San Antonio de los Cobres (SAC ; Argentine)	24.05 S 66.24 W	3600	20.9	-8.9
Meteor Crater (USA)	35.04 N 111.03 W	1680	23.6	42.7
San Pedro Martir (SPM ; Mexique)	31.01 N 115.48 W	2400	25.3	38.4
Teide, Tenerife (Espagne)	28.28 N 16.54 W	2290	30.8	23.2
Aar@500 m (site hypothétique)	26.69 S 6.44 E	500	10.9	-24.9

Les caractéristiques principales des différents sites candidats sont résumées dans la Table 3.1. Une grande statistique d'événements a été simulée afin de calculer les fonctions de réponse pour chaque site candidat. Les performances obtenues pour chaque site sont exprimées en termes de sensibilité différentielle du réseau à une source ponctuelle. Cette sensibilité est définie comme le flux minimum détectable en provenance d'une source ponctuelle stable, calculé sur une bande d'énergie réduite. En complément, les performances en termes de résolution angulaire ont été considérées.

Les résultats de l'analyse des simulations montrent que les performances d'un réseau de type CTA varient fortement d'un site à l'autre, les paramètres importants étant l'altitude et le champ magnétique orthogonal. Les meilleures performances globales sur l'étendue du spectre d'énergie seraient attendues pour une altitude comprise entre 1600 m et 2500 m. Ces résultats ont fait l'objet de deux présentations à l'ICRC 2015 [32] [33] et les conclusions définitives ont été publiées dans *Astroparticle Physics* en 2017 [34].

Ces études ont fourni des indicateurs de performance essentiels pour le processus de sélection des sites par l'observatoire. Cependant d'autres critères ont été également pris en compte, comme le coût des infrastructures, les risques en phase de construction ou lors des observations. Au final, les deux sites choisis, Paranal (proche d'Armazones) et La Palma (proche du Teide, aux îles Canaries) remplissent globalement les critères issus de cette étude en terme d'altitude et de champ magnétique.

3.4.4.2 Choix de la géométrie

Une fois les sites sélectionnés, la PROD3 a été lancée (2015/2016) avec l'objectif d'étudier les performances de CTA en fonction de la géométrie des réseaux. Plusieurs facteurs ont été considérés, tels que la forme du réseau (carré ou hexagonale), l'espacement entre télescopes, la présence d'îles ou de sous réseaux indépendants. Les détails de cette étude sont disponibles dans l'article [35] publié dans *Astroparticle Physics*.

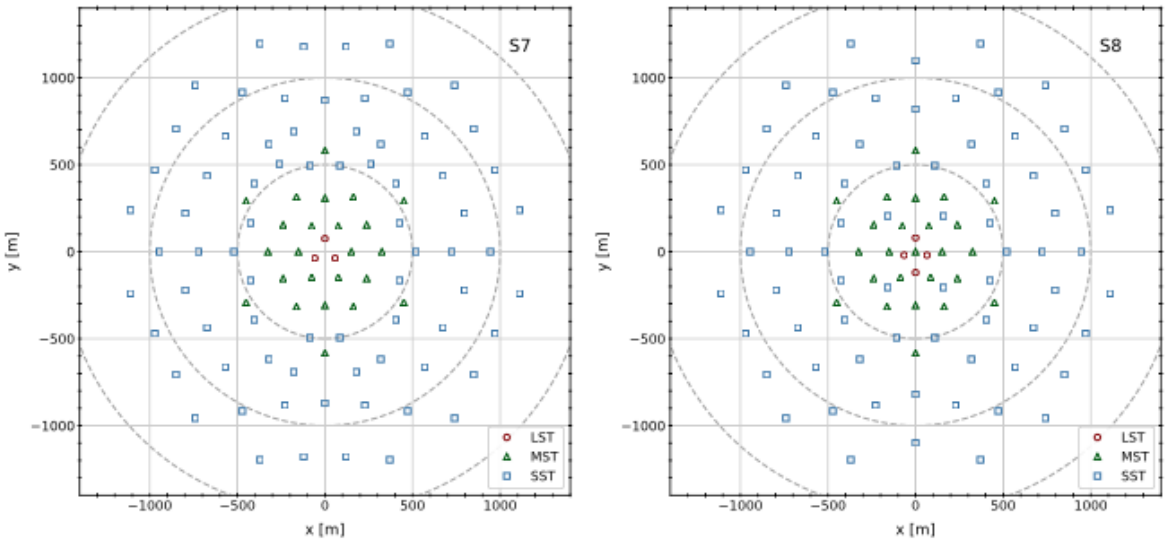

FIGURE 3.5 – À gauche, le réseau S7 au site Sud, avec lequel on obtient les meilleures performances. À droite, le réseau final proposé (S8), dérivé du S7 en ajustant la position de quelques télescopes [35].

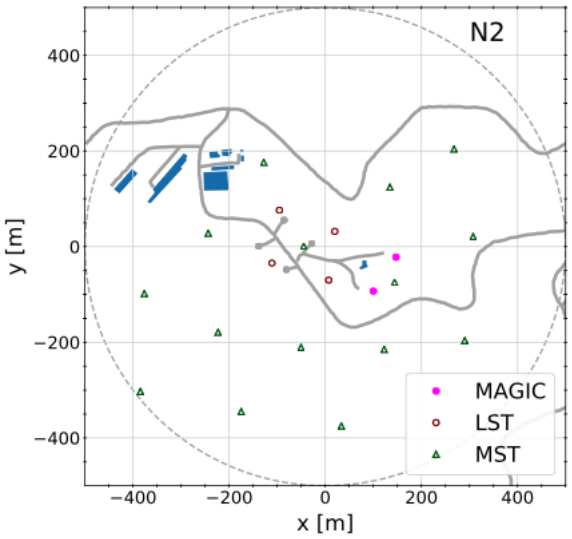

FIGURE 3.6 – Réseau final proposé (N2) au site Nord. Les routes sont indiquées en gris, les bâtiments en bleu et les télescopes de MAGIC en magenta [35].

La stratégie d'optimisation a été établie sur des bases similaires à celles pour le choix des sites. Le programme scientifique de CTA étant très large, il s'agit de trouver les meilleures performances globales, sans favoriser une gamme d'énergie plutôt qu'une autre et sans favoriser la surface effective au détriment de la résolution angulaire.

Afin de comparer différentes configurations, nous avons utilisé un réseau fictif comprenant 800 positions. De plus, pour chaque type de télescope, nous avons simulé les différents modèles de caméras disponibles (1 pour les LST, 2 pour les MST et 3 pour les SST) pour un total de 3092 instruments. Du fait du nombre élevé de télescopes, les tâches de simulations présentaient une consommation de la mémoire vive et une durée particulièrement importante par rapport à une simulation standard. Un travail d'optimisation a donc été nécessaire afin de pouvoir utiliser la vingtaine de sites de la grille à disposition.

Pour l'analyse, les 3 chaînes d'analyses mentionnées dans la Section 3.4.1 ont été utilisées, ce qui a permis de montrer la robustesse des résultats vis-à-vis du logiciel utilisé. La première partie de cette étude, réalisée au MPIK par K. Bernlöhr, a mis en évidence une légère préférence pour une géométrie hexagonale par rapport à une géométrie carrée. En se focalisant donc sur des géométries hexagonales, une deuxième vérification a consisté à montrer que les diverses combinaisons possibles de télescopes et caméras n'ont pas d'impact sur les tendances des performances en passant d'une géométrie à l'autre.

En outre, une partie importante de cette étude a consisté à faire varier l'espacement entre télescopes et étudier ainsi l'impact de l'emprise au sol du réseau sur les performances globales. Pour cela un facteur d'échelle homothétique a été appliqué à la distance entre télescopes, avec 5 valeurs différentes allant de 1 à 5. Tel qu'attendu, plus le réseau est étendu et plus la surface effective augmente à haute énergie. En revanche, lorsque les télescopes sont trop distants les uns des autres, la multiplicité (nombre de télescopes 'touchés' par un événement) chute, et la reconstruction des événements perd en qualité.

Pour le site Sud, après une étude détaillée, les meilleures performances ont été obtenues en combinant un réseau de MST avec un facteur de mise à l'échelle de 2 et un réseau de SST avec un facteur de 5 (réseau S7). Le réseau final (S8), représenté sur la Figure 3.5, a été obtenu en ajustant la position de quelques MST pour favoriser l'inter-étalonnage avec les SST.

Pour le choix final du réseau au site Nord de La Palma, des contraintes supplémentaires ont été prises en compte, en raison en particulier des fortes pentes du volcan, de la présence d'espaces naturels protégés et de nombreux autres télescopes. Le réseau de référence proposé est représenté sur la Figure 3.6.

3.4.5 Ressources de calcul

Les ressources de calcul mises en jeu et les moyens humains employés pour la production des simulations Monte-Carlo présentées dans les sections précédentes sont très importants. À titre d'exemple, la PROD3 seule a nécessité sur la grille 123.6 millions d'heures HS06 [1] de CPU et 1.4 Po d'espace disque, pour un coût total estimé à 500k euros, hors salaire des chercheurs et ingénieurs impliqués. Il est aisé d'entrevoir les enjeux économiques importants de l'optimisation de l'utilisation des ressources de calcul.

1. `https://w3.hepix.org/benchmarking.html`

Au total, le CPU consommé pour les différentes productions effectuées entre janvier 2013 et novembre 2020 s'élève à environ 760 millions d'heures HS06, soit presque 100 millions d'heures par an (cf. Figure 3.7). Le nombre total de tâches exécutées pendant cette même période est de 17.2 millions, avec des pics d'environ 10k tâches en simultané lors des phases de production intensives. L'espace de stockage occupé par ces simulations est également significatif. Actuellement environ 4.5 Po de données sont reparties entre six sites de la grille, en partie sur disque et en partie sur bande magnétique. En outre, des transferts de données entre sites sont automatiquement déclenchés par les tâches, lors de l'écriture des résultats de sortie et lors de l'accès aux données d'entrée. Plusieurs Po par an sont ainsi transférés via le réseau avec un débit global de plus de 1 Go/s (cf. Figure 3.7).

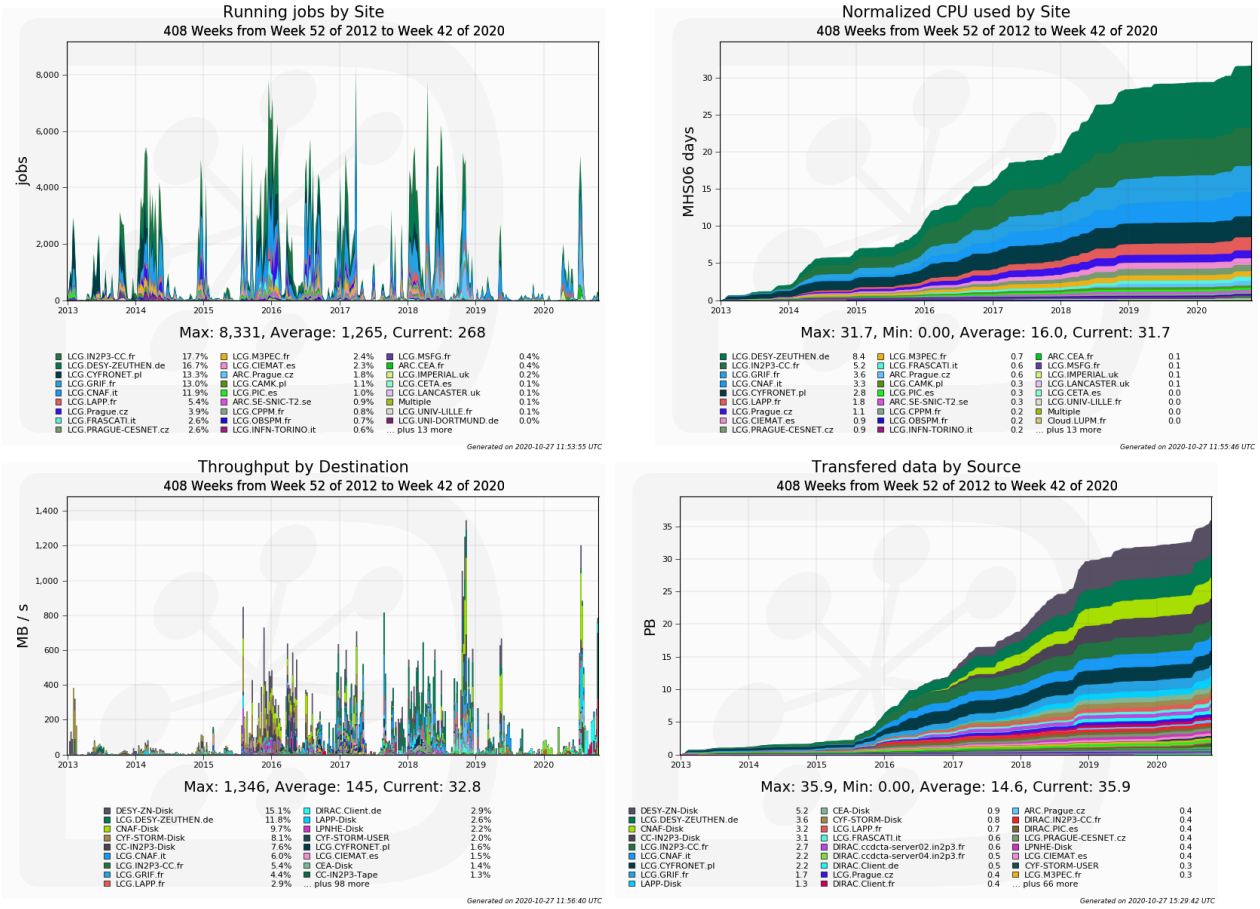

FIGURE 3.7 – Utilisation des ressources de la grille pour la production et l'analyse des simulations Monte-Carlo de CTA entre janvier 2013 et novembre 2020. En haut à gauche, nombre de tâches concurrentes reparties sur les différents sites. En haut à droite, distribution cumulée du nombre d'heures CPU normalisées. En bas à gauche, débit d'accès sur disque en fonction des sites de destination. En bas à droite, distribution cumulée de la quantité de données transférée depuis chaque site.

3.5 Conclusions

La production des simulations Monte-Carlo sur la grille a constitué pour nous une excellente opportunité d'évaluer de manière approfondie les différentes fonctionnalités de DIRAC dans un contexte d'application réelle. Nous avons utilisé le framework DIRAC autant pour la gestion des tâches (WMS) que pour la gestion des données (DMS). En particulier, nous avons utilisé le WMS pour accèder aux ressources d'environ une vingtaine de sites de la grille. Grâce à la flexibilité du WMS, nous avons pu facilement configurer l'instance `CTA-DIRAC` pour interfacer les différents types de *Computing Element* (CREAM, ARC, HT-Condor, etc.) apparus au cours des dernières années. Dans le cadre du projet HNSciCloud sur l'utilisation des clouds publics et privés, j'ai également testé de manière extensive le module VMDIRAC, en obtenant des résultats satisfaisants [26]. Enfin, j'ai eu également l'occasion d'intégrer des fermes de calcul 'hors grille' en connectant directement le système de batch.

Ces caractéristiques fondamentales du WMS de DIRAC de permettre l'intégration de ressources hétérogènes et d'être facilement extensible sont particulièrement importantes dans le contexte d'un projet de longue durée comme CTA. Au niveau du passage à l'échelle, l'instance `CTA-DIRAC` a supporté sans problèmes des pics d'environ 10k tâches en simultané. Une utilisation encore plus intensive du WMS est obtenue par l'expérience LHCb, qui gère à régime environ 60k tâches en simultané distribuées sur près de 120 sites.

La gestion automatisée des *workflows* est une autre fonctionnalité qui intéresse CTA. En effet, l'observatoire se doit de produire des données de qualité ouvertes au public. L'automatisation des *workflows* permet non seulement de réduire les coûts en ressource humaines pour la gestion des productions, mais contribue également à la qualité du processus de production de données. Dans ce contexte, à partir de la PROD3 j'ai commencé à évaluer le *Transformation System* de DIRAC pour la gestion des *workflows*. Comme nous l'avons vu dans la Section 3.4.2, le nombre de transformations nécessaires pour une campagne de simulation complète est relativement élevé, ce qui rend difficile leur monitorage. De plus, la création manuelle de toutes ces transformations est une source d'erreur non négligeable. Le *Production System* que j'ai développé est une couche logicielle entre le *Transformation System* et l'utilisateur, qui ajoute le degré d'automatisation nécessaire à la gestion des *workflows*. J'ai testé ce nouveau système lors de la dernière production (PROD5) et vérifié qu'il répond bien aux besoins d'automatisation de CTA.

De manière analogue à la gestion des ressources de calcul par le WMS, le système de gestion de données (DMS) de DIRAC permet d'interfacer différents types de système de stockage de manière transparente pour l'utilisateur. Encore une fois, cette fonctionnalité est nécessaire dans le cadre de projets utilisant des ressources distribuées, comme CTA. Grâce au DMS de DIRAC, nous avons géré efficacement plusieurs Po de données de simulation distribuées sur six sites. Toutes ces données sont également référencées dans le catalogue de DIRAC (DFC) qui contient actuellement environ 20 millions de fichiers. Afin de caractériser ces données, nous avons défini dans le DFC une vingtaine de métadonnées. Ces dernières servent en particulier à définir les filtres des transformations qui composent les *workflows* de simulation. Lors des productions, plusieurs milliers de tâches concurrentes exécutent de multiples requêtes au DFC (enregistrement, accès aux fichiers, mise à jour des valeurs des métadonnées, etc.). De plus, le WMS et le *Transformation System* interrogent constamment le DFC pour l'optimisation de l'ordonnancement et l'association des fichiers aux transforma-

tions. Au cours de cette utilisation intensive du DFC, nous n'avons observé aucune limitation en termes de performances.

À la vue de ces résultats, le LUPM a l'objectif de proposer le prototype `CTA-DIRAC` comme *Inkind Contribution* à l'observatoire pour le gestionnaire de tâches de CTA (`CWMS`). Pour la partie archivage, nous avons vérifié que le DMS de DIRAC possède un certain nombre de fonctionnalités nécessaires à CTA, i.e. capacité d'intégrer des systèmes hétérogènes et distribués, support d'opérations massives sur les données, fonctionnalité de catalogue, etc. Néanmoins l'évaluation de toutes les fonctionnalités requises par l'observatoire et le développement d'un prototype complet demandent encore du travail. En outre, d'autres technologies sont également en cours d'évaluation au sein du consortium, telles que *Rucio* issu de l'expérience ATLAS et *Onedata* développé dans le cadre du projet européen eXtremeDataCloud. La décision finale sur la technologie à adopter dépendra donc du résultat de ces évaluations en termes de fonctionnalités, mais également d'autres critères, comme le support sur le long terme ou la possibilité de partager la même solution avec d'autres observatoires.

Chapitre 4

Optimisation du code de simulation de cascades atmosphériques CORSIKA

Dans les chapitres précédents, nous avons abordé la problématique de l'optimisation de l'utilisation des ressources de calcul du point de vue de l'ordonnancement et de l'orchestration des tâches. Les systèmes de WMS optimisent l'utilisation des ressources en réduisant le délai entre la soumission et l'exécution des tâches et en maximisant le taux d'occupation des ressources à disposition. De manière complémentaire, pour que les ressources ainsi réservées soient utilisées efficacement, les applications doivent exploiter au mieux les capacités des processeurs. Dans ce dernier chapitre, je vais précisément aborder la problématique de l'optimisation de code avec les contraintes de portabilité dues à l'utilisation d'infrastructures hétérogènes comme celle de la grille.

Plus particulièrement, dans le cadre du projet PEPS Astro-Informatique `CTAOptSim` (voir Annexe A.2), je me suis intéressée à l'optimisation du programme de simulation Monte-Carlo de cascades atmosphériques `CORSIKA`, utilisé par CTA et de nombreuses autres communautés. Une des motivations principales du choix de `CORSIKA` pour ce travail d'optimisation, réside dans le fait que les simulations représentent la majeure partie du temps CPU global consommé par CTA. Ce même constat d'ailleurs s'applique de manière générale à toutes les expériences en physique des hautes énergies. En outre, `CORSIKA` fait partie de la catégorie des codes Monte-Carlo dits de 'transport', tels que le bien connu framework `Geant` pour la simulation du passage des particules à travers des détecteurs. Le travail sur `CORSIKA` nous permet donc de gagner en compétence sur des problématiques d'optimisation qui sont communes aux codes de transport.

Au niveau de l'IN2P3, le projet `Reprises` a été lancé en 2017 à l'initiative de plusieurs laboratoires (voir Annexe A.3) avec l'objectif de traiter, de manière transversale aux expériences, les thématiques de performance, portabilité et précision du calcul. Je participe à `Reprises` depuis sa création, en lien avec mon activité sur `CORSIKA` et depuis le printemps 2020 j'ai repris la coordination du projet.

Dans ce chapitre, je vais présenter les grandes lignes du travail effectué sur l'optimisation de `CORSIKA`. La première partie du travail a été réalisée par les partenaires de `CTAOptSim` et les résultats ont été présentés à la conférence CHEP [36]. Le projet s'est ensuite poursuivi dans le cadre du stage de Master d'Adnane Khattabi, que j'ai co-encadré avec David Parello (Maître de conférences à l'Université de Perpignan et membre de l'équipe DALI du LIRMM),

cf. Annexe B.1. Ces deux premières phases du projet sont décrites dans les Sections 4.4 et 4.5. Enfin, le travail d'optimisation a été repris dans le cadre de la thèse de Matthieu Carrère, qui a démarré en octobre 2019 et que je co-encadre avec D. Parello (cf. Annexe B.2). Les travaux présentés dans cette section ont fait également l'objet d'une publication dans *Computing and Software for Big Science* [37], où l'on peut trouver de plus amples explications.

4.1 Motivations

Comme nous l'avons vu dans la Section 1.4.2, les simulations nécessitent des ressources de calcul très importantes. Pendant la phase de préparation de CTA, environ 100 millions d'heures HS06 par an ont été consommées pour les simulations dédiées à l'optimisation du réseau. Lors de la future phase d'opération, des simulations seront effectuées pour chaque observation afin de produire des fonctions de réponse les plus précises possibles.

La réduction du temps CPU des simulations est donc un enjeu majeur pour CTA. Cela permettrait soit de produire la statistique nécessaire en moins de temps en réduisant ainsi les coûts, soit de produire une statistique plus élevée et réduire les erreurs statistiques au même prix.

Nous rappelons que le pipeline de simulation repose sur le logiciel `CORSIKA` pour la simulation des cascades et sur le logiciel `sim_telarray` pour la simulation de la réponse des télescopes. Le temps de calcul global est reparti à hauteur de 70% pour `CORSIKA` et 30% pour `sim_telarray`. C'est pourquoi nous avons concentré nos efforts sur `CORSIKA` et en particulier sur la version actuelle de référence (`CORSIKA 7`).

Nous rappelons également qu'un projet de ré-écriture complète de `CORSIKA` en C++ moderne (projet `CORSIKA 8` [1]) est actuellement en cours. Bien que le développement de `CORSIKA 8` soit bien avancé, il faudra un certain temps avant d'obtenir une version stable et validée. CTA pourra donc bénéficier du travail d'optimisation sur `CORSIKA 7` encore plusieurs années.

4.2 Le logiciel CORSIKA

`CORSIKA` consiste d'un module principal, écrit en Fortran, qui gère la pile des particules, le transport des particules, la génération de nombres aléatoires et la description de l'atmosphère. La simulation des cascades électromagnétiques est incluse dans le code principal en utilisant une version adaptée du programme EGS4. Les modèles d'interaction hadroniques sont implémentés dans plusieurs modules externes, également en Fortran, utilisés comme des plugins par le programme principal.

Les expériences à imagerie Cherenkov, comme CTA, utilisent un module supplémentaire `iact/atmo`, écrit en C, qui gère la génération des photons Cherenkov dans les cascades, la propagation de ces photons à travers l'atmosphère et la géométrie en 3D de réseaux de télescopes. Le module `iact/atmo` supporte en outre une description de l'atmosphère basée sur des tables externes (densité, indice de réfraction en fonction de l'altitude, etc.), qui est beaucoup plus précise que celle fournie par défaut dans `CORSIKA`, basée sur un modèle analytique.

1. `https://www.iap.kit.edu/corsika/`

Comme dans tout code de transport, la simulation d'une cascade dans `CORSIKA` se base sur le suivi des particules produites dans la cascade. En traversant l'atmosphère, les particules subissent une série d'interactions dues à différents processus physiques. Pour les expériences à imagerie Cherenkov, l'atmosphère représente donc le détecteur. Entre deux interactions, une particule est 'transportée' d'un point à un autre en parcourant un *step*. Chaque particule est ainsi représentée par une trace composée d'un certain nombre de *steps*. L'émission et la propagation Cherenkov sont traitées suivant les étapes illustrées sur la Figure 4.1. Pour chaque *step*, le nombre de photons Cherenkov émis est calculé. Afin de réduire le temps de calcul, ces photons sont regroupés en *bunch* de typiquement 5 photons. Chaque *step* est subdivisé ultérieurement en *sub-steps* de sorte qu'un seul *bunch* de photons soit émis à chaque *sub-step*. Ensuite, chaque *bunch* est propagé à travers l'atmosphère en prenant en compte la déviation de sa trajectoire par effet de la réfraction. Enfin, les *bunch* qui arrivent jusqu'aux

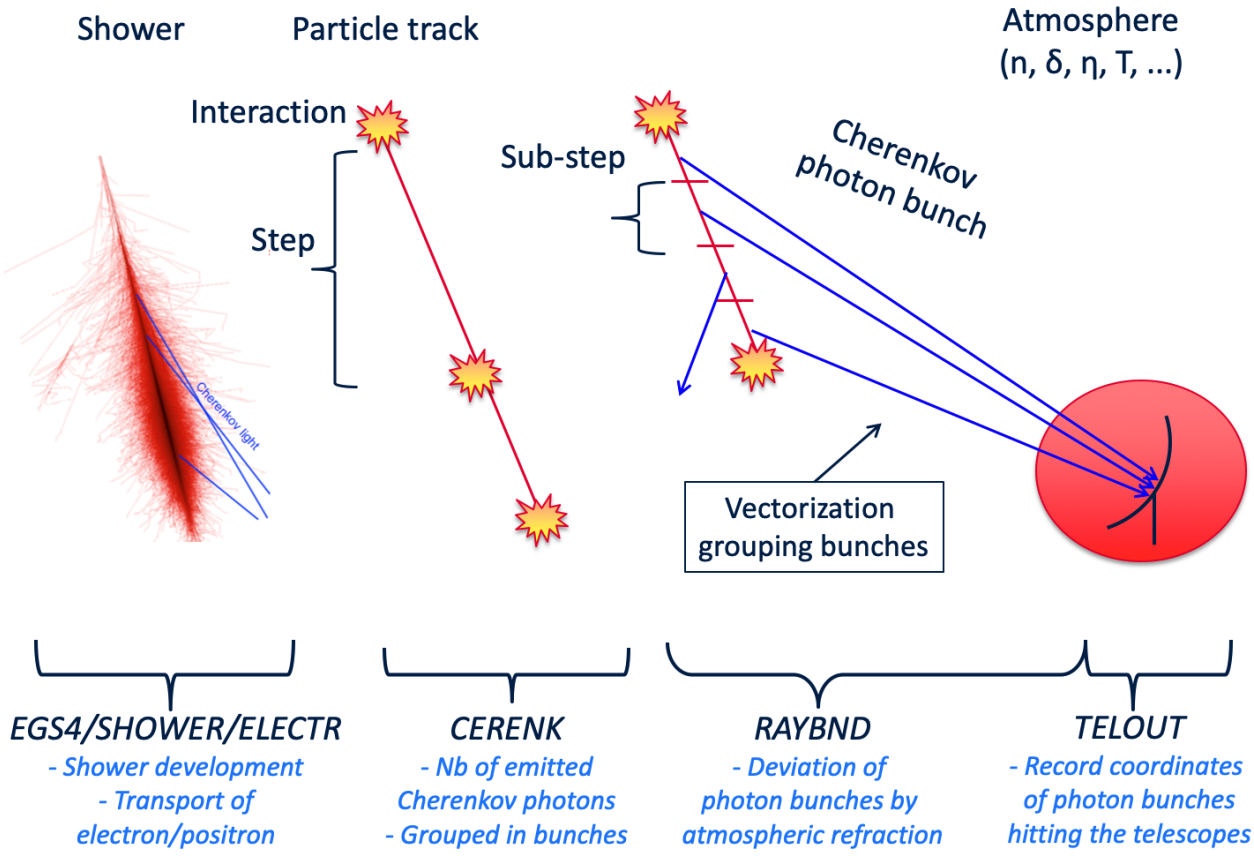

FIGURE 4.1 – Vue schématique des différentes routines qui interviennent dans la simulation d'une cascade électromagnétique. Le développement de la gerbe et le transport des électrons et positrons sont gérés par les routines `EGS4`, `SHOWER` et `ELECTR`. Pour chaque *step*, la routine `CERENK` calcule le nombre de photons Cherenkov émis et le nombre de *bunch* correspondant. La routine `RAYBND` est en charge de la propagation des *bunch* en prenant en compte l'effet de la réfraction atmosphèrique. Enfin, la fonction `TELOUT` vérifie quels *bunch* arrivent jusqu'aux télescopes et les enregistrent dans un fichier de sortie.

télescopes sont enregistrés dans un fichier de sortie. Les télescopes sont représentés par des sphères à différentes positions, ce qui permet de simuler un réseau. La simulation détaillée de la réponse optique et de l'électronique des caméras des télescopes est ensuite implémentée dans le logiciel `sim_telarray`.

4.3 Stratégie d'optimisation

En fonction du cas d'utilisation de `CORSIKA`, il existe plusieurs méthodes pour réduire le temps de calcul. Cela dépend fortement des processus physiques prépondérants et de la gamme d'énergie considérée. Dans la gamme d'énergie de CTA (30 GeV - 300 TeV), la simulation d'une cascade individuelle (par la suite appelée également 'événement') induite par un photon gamma est relativement rapide, avec une durée allant de quelques secondes à quelques minutes, pour les cascades les plus énergétiques. Toutefois, afin d'étudier les fluctuations d'une cascade à une autre, il est nécessaire de produire une statistique très élevée. Chaque production consiste typiquement de quelques milliards d'événements simulés correspondant à environ 60 millions d'heures CPU HS06. Ces événements étant tous indépendants, la production de la statistique complète est obtenue en distribuant un grand nombre de tâches `CORSIKA` sur plusieurs coeurs. Chaque tâche simule environ 50k cascades en quelques heures. Dans le modèle actuel (voir Section 3.2), le consortium effectue une production en utilisant entre 8k et 10k coeurs en simultané sur la grille en une période d'un mois.

Dans d'autres expériences, comme Pierre Auger Observatory, les gammes d'énergies considérées sont plus élevées. Dans ce cas, la limitation principale vient du temps CPU nécessaire à la simulation d'événements très énergétiques (au delà de 10^{17} eV). À titre d'exemple, la simulation d'une seule cascade induite par un hadron de 10^{18} eV requière un mois de CPU et environ 100 Go d'espace disque.

Plusieurs méthodes ont été développées pour accélérer la simulation de ces événements ultra énergétiques. Ces méthodes consistent à appliquer différents types d'approximations, comme la méthode de *thinning* [38] ou la méthode utilisant des solutions numériques des équations des cascades. Enfin, une alternative consiste à paralléliser `CORSIKA` en distribuant la simulation d'une cascade sur plusieurs coeurs via le protocole MPI.

Dans le cas de CTA, le facteur limitant étant la très grande statistique d'événements plutôt que les événements ultra énergétiques, il n'y a pas de véritable intérêt à utiliser la version parallélisée de `CORSIKA`. Notre approche pour accélérer les simulations consiste donc à réduire le temps d'exécution séquentielle. Pour atteindre cet objectif, nous avons exploré différentes techniques et en particulier les techniques de vectorisation. L'objectif final est d'améliorer les performances sans impacter la précision des résultats. Une contrainte supplémentaire concerne la portabilité du code, qui doit pouvoir s'exécuter sur les différents centres de calcul de la grille. Enfin, nous n'avons pas considéré la possibilité de décharger une partie des calculs sur GPUs, du fait de leur faible disponibilité dans le cadre actuel de CTA.

4.3.1 Principe de la vectorisation

Le principe de la vectorisation consiste à utiliser des instructions SIMD (*Single Instruction on Multiple Data*) pour effectuer une même opération sur plusieurs données à la fois, tel que

schématisé sur la Figure 4.2. Les implémentations plus répandues des instructions SIMD sont : SSE4, AVX/AVX2, AVX–512, opérant respectivement sur des registres de 128, 256 et 512 bits. Cela signifie qu'en utilisant par exemple des instructions AVX2, une opération donnée sera exécutée en parallèle sur 4 nombres en double précision ou sur 8 nombres en simple précision. Le gain théorique maximal apporté par la vectorisation est donc respectivement égal à un facteur 4 ou 8.

En outre, un des avantages de la vectorisation est que les instructions SIMD sont d'ores et déjà disponibles sur tous les processeurs modernes, cf. Table 4.1. Nous pouvons donc bénéficier immédiatement de la vectorisation sur les centres de calcul de la grille.

Néanmoins, ces centres sont équipés avec différents modèles de CPU ayant différents niveaux de support des instructions SIMD. Afin d'obtenir un code vectorisé portable sur cette infrastructure, nous avons choisi l'approche de l'auto-vectorisation en opposition à l'utilisation de fonctions de bas niveau (intrinsèques). En effet, le problème de l'utilisation des fonctions intrinsèques est qu'elles sont spécifiques à une architecture donnée. En revanche l'auto-vectorisation est la vectorisation obtenue automatiquement par le compilateur en activant certaines options. La limite principale de cette approche est que le compilateur est capable de vectoriser un code seulement sous certaines conditions. Nous verrons dans la Section 4.5, les principales transformations du code qui ont été nécessaires pour permettre l'auto-vectorisation de `CORSIKA`.

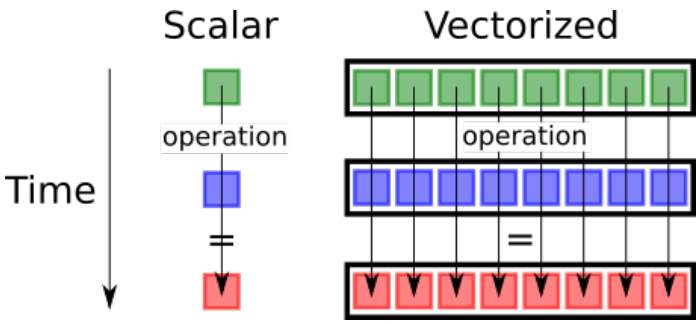

FIGURE 4.2 – Principe de la vectorisation. À gauche, une opération scalaire. À droite, une opération vectorielle appliquée simultanément à plusieurs données.

4.4 Profilage

Afin d'orienter le travail d'optimisation, nous avons d'abord effectué plusieurs profilages de `CORSIKA` et identifié les fonctions qui consomment la majeure partie du temps CPU. La Figure 4.3 illustre le résultat du profilage obtenu avec *gprof* pour une exécution avec des paramètres utilisés habituellement dans les simulations de CTA. On observe que plus de 90% du temps CPU est consommé dans la routine `CERENK`, responsable de la génération des photons Cherenkov et 50% dans la routine `RAYBND`, en charge de la propagation des *bunch* Cherenkov à travers l'atmosphère. Ceci s'explique entre autre par le fait que ces routines sont très fréquemment appelées, le nombre de photons Cherenkov produits dans une cascade étant très élevé (670k par cascade en moyenne dans cette exécution). En outre, le profilage avec

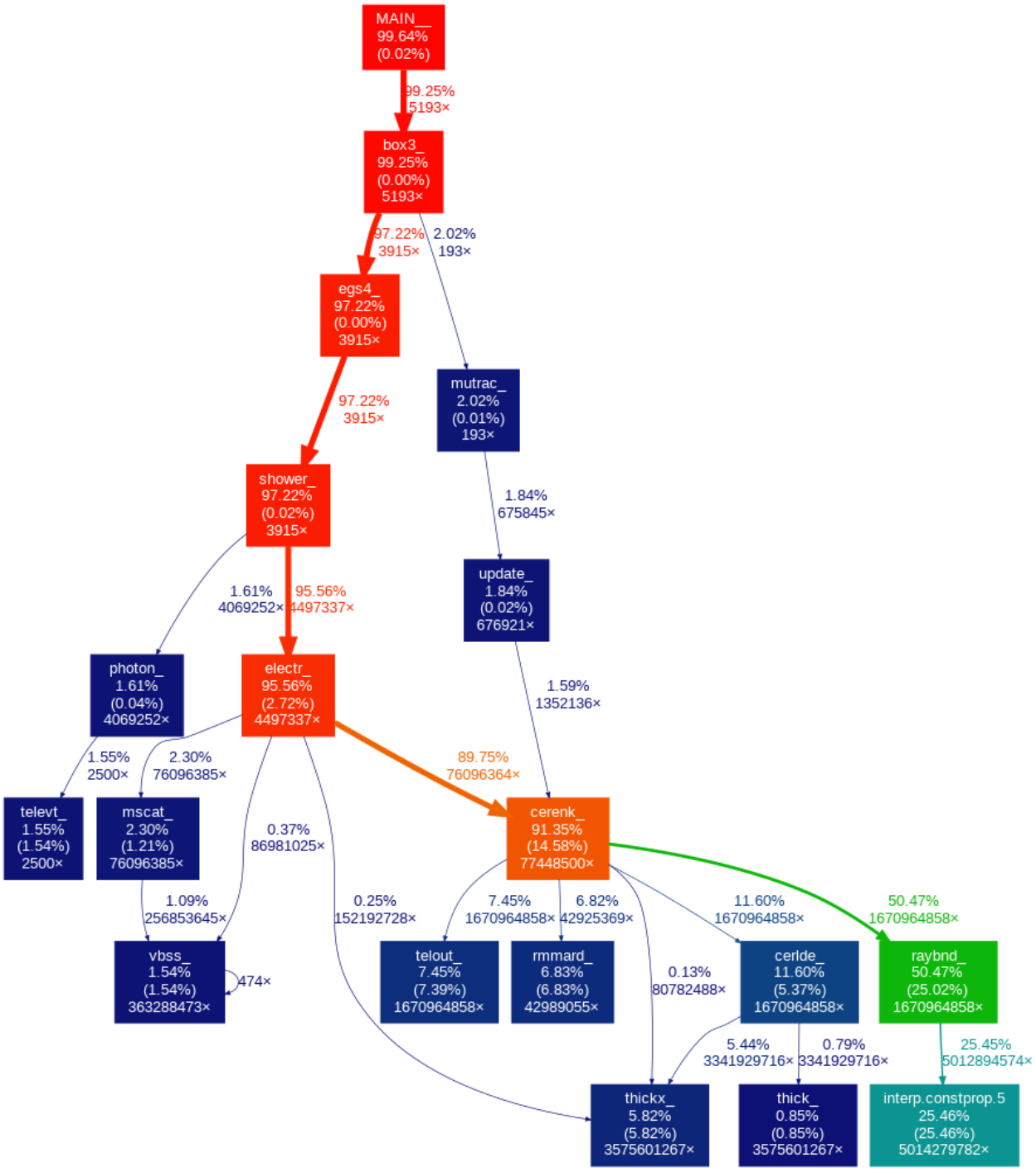

FIGURE 4.3 – Arbre d'appels généré avec *gprof* et *gprof2dot* pour la simulation de 2500 cascades avec `CORSIKA`. Le pourcentage de consommation CPU de chaque routine est indiqué avec une dégradation du rouge au bleu pour les valeurs des plus élevées aux plus faibles.

TABLE 4.1 – Evolution des capacités de vectorisation des processeurs Intel [39].

Instruction Set	Release Date	First CPU	Nb **float** per instruction	Nb **int** per instruction
SSE2	2003	2004	2	1
SSSE3	2004	2005	2	2
SSE4.1	2006	2007	4	2
SSE4.2	2007	2008	4	4
AVX	2008	2011	8	4
AVX2	2012	2013	8	8
AVX 512.1	2013	2016	16	8
AVX 512.2	2015	2018	16	16

perf, qui inclut également les librairies de bas niveau, montre qu'une fraction importante du CPU est utilisée par des fonctions élémentaires (*exp*, *asin*, *cos*, *sin*) [36].

Enfin, puisque les photons Cherenkov produits dans les cascades sont indépendants les uns des autres, chaque *bunch* est également propagé de manière indépendante dans `RAYBND`. Nous avons donc étudié la possibilité de vectoriser `RAYBND` afin de traiter plusieurs *bunch* à la fois et réduire ainsi le temps d'exécution. Puisque tous les calculs dans `CORSIKA` sont en double précision, sur un processeur avec de l'AVX2, on pourrait théoriquement réduire d'un facteur 4 le temps CPU passé dans `RAYBND`.

4.5 Optimisation par la vectorisation

La routine `RAYBND` propage chaque *bunch* à travers l'atmosphère en prenant en compte l'effet de la réfraction. Chaque *bunch* étant émis dans un *sub-step*, `RAYBND` est appelée dans la routine `CERENK` dans une boucle sur les *sub-steps*. Les paramètres passés à `RAYBND` sont les coordonnées du *bunch* (position, direction, temps d'arrivée, etc.) à propager.

Afin de vectoriser `RAYBND`, il a été tout d'abord nécessaire de transformer la boucle sur les *sub-steps* afin de passer à `RAYBND` des vecteurs contenant les coordonnées des *bunch* appartenant à des *sub-steps* consécutifs. La taille de ces vecteurs est un paramètre qu'on ajuste à la compilation, et dont nous avons trouvé une valeur optimale égale à 8.

Après cette transformation, nous avons écrit une version vectorisée de `RAYBND`. Il s'agit de transformer `RAYBND` pour que les opérations s'effectuent de manière vectorielle sur les éléments des vecteurs passés en paramètre. Plus précisément, nous avons écrit les noyaux de calcul dans des boucles opérant sur les éléments de ces vecteurs. Ensuite, la partie plus délicate du travail a consisté à restructurer le code afin d'isoler ces boucles pour que le compilateur puisse les identifier et les vectoriser. Une analyse détaillée des rapports de compilation a été nécessaire pour comprendre dans quelles conditions le compilateur n'arrive pas à vectoriser. Tel qu'attendu, nous avons observé que les branchements présents dans le code empêchaient la vectorisation automatique. De plus, lorsque des instructions contiennent des appels à des fonctions, celles-ci ne sont pas vectorisées. Ce dernier schéma apparaît fréquemment dans

RAYBND pour effectuer des interpolations des profils d'atmosphère. Nous avons donc restructuré le code, d'une part en regroupant les instructions conditionnelles pour minimiser les branchements et d'autre part en éliminant les appels à des fonctions dans les instructions à vectoriser.

Enfin, le profilage de RAYBND montre qu'une fraction importante du temps CPU est utilisée par des fonctions élémentaires. La propagation des *bunch* implique notamment beaucoup d'appels à des fonctions trigonométriques. Pour vectoriser ces fonctions, nous avons utilisé une librairie externe dédiée. Plusieurs librairies de fonctions élémentaires vectorisées sont actuellement disponibles avec différentes caractéristiques en termes de performances, précision, portabilité et disponibilité du code (*open-source* ou non). Après une étude comparative [36], nous avons décidé d'adopter la librairie *open-source* 'vector libm' [40] qui présente des bonnes performances, une précision suffisante et qui est portable sur différentes architectures. Le gain supplémentaire obtenu par l'utilisation de cette librairie est significatif, de l'ordre de 20%. Il est à noter que les fonctions élémentaires fournies par ces librairies opèrent le plus souvent sur des vecteurs. Une restructuration du code, comme celle présentée ci-dessus, est donc un pré-requis nécessaire pour pouvoir les utiliser.

Après toutes ces transformations, nous avons mesuré le taux global de vectorisation et nous l'avons comparé au taux de la version originale. Ces mesures indiquent un taux de vectorisation sur l'ensemble du code d'environ 60%, à comparer avec un taux presque nul dans la version originale (cf. Figure 4.4).

L'accélération globale (*speed-up*) obtenue avec la version optimisée sur un processeur avec de l'AVX2 est égale à 1.5. Compte tenu du fait que RAYBND représente environ 50% du temps CPU global, cette accélération est compatible avec une accélération de 4 sur RAYBND, qui est le maximum théoriquement attendu pour un code en double précision sur de l'AVX2.

4.6 Optimisations ultérieures

Le travail d'optimisation présenté ci-dessus, a ensuite été repris dans le cadre de la thèse de Matthieu Carrère, démarrée en octobre 2019. Matthieu a exploré des pistes complémentaires d'optimisation en étudiant notamment les accès mémoire. Grâce à cette analyse, nous avons identifié un problème fondamental dans le code qui générait un taux très élevé de défauts de cache. Environ 71% des défauts de cache sont en effet générés par les appels à une fonction particulière ($memset\_sse2$), comme indiqué sur la cf. Figure 4.5. Par chance, il a été facile de corriger ce problème car la fonction $memset\_sse2$ est à son tour appelée par une fonction bien identifiée, i.e. la fonction TELOUT (cf. Figure 4.1). La fonction TELOUT appelle $memset\_sse2$ via un appel à la fonction *calloc* pour l'initialisation des blocs mémoire alloués pour les structures des *bunch*. Après avoir vérifié que l'initialisation des blocs mémoire n'était pas nécessaire dans le cas présent, il a donc suffit de remplacer la fonction *calloc* par la fonction *malloc* pour réduire drastiquement le taux de défauts de cache. Le gain de performance global qui en a découlé est assez conséquent, l'accélération passant de 1.5 à 1.7.

Un autre paramètre ayant un impact sur la vectorisation et sur le nombre d'appels aux fonctions est la taille des vecteurs passés en paramètre dans la version vectorisé de RAYBND. Matthieu a étudié l'impact de ce paramètre, initialement fixé à 4, et trouvé des performances maximales pour une valeur égale à 8. Grâce à cet ajustement, la version finale optimisée de

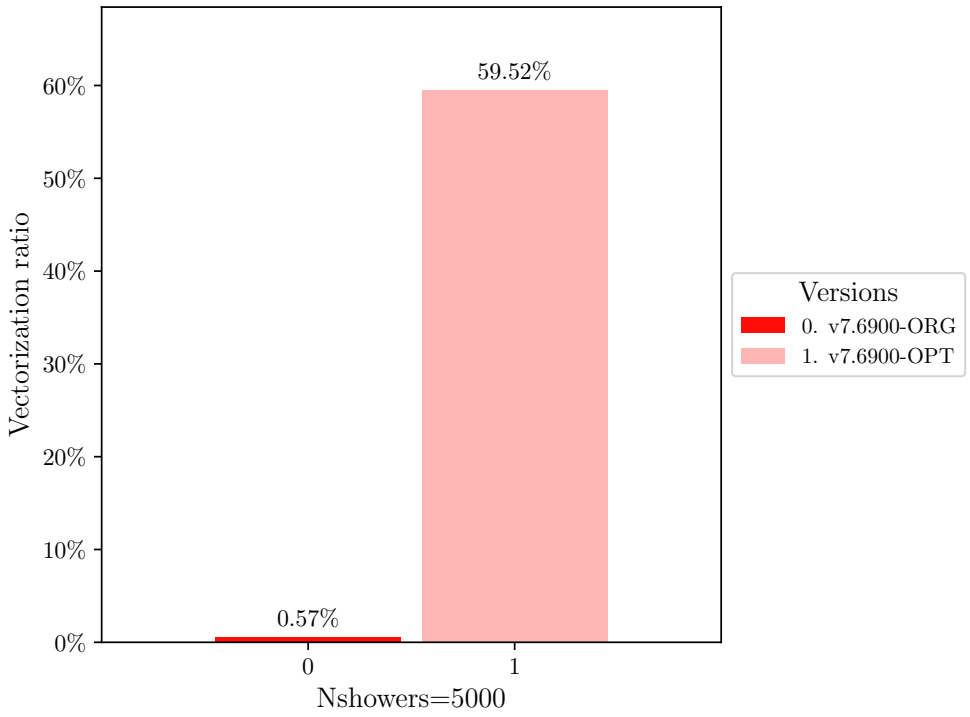

FIGURE 4.4 – Taux de vectorisation dans la version originale (à gauche) et dans la version optimisée (à droite).

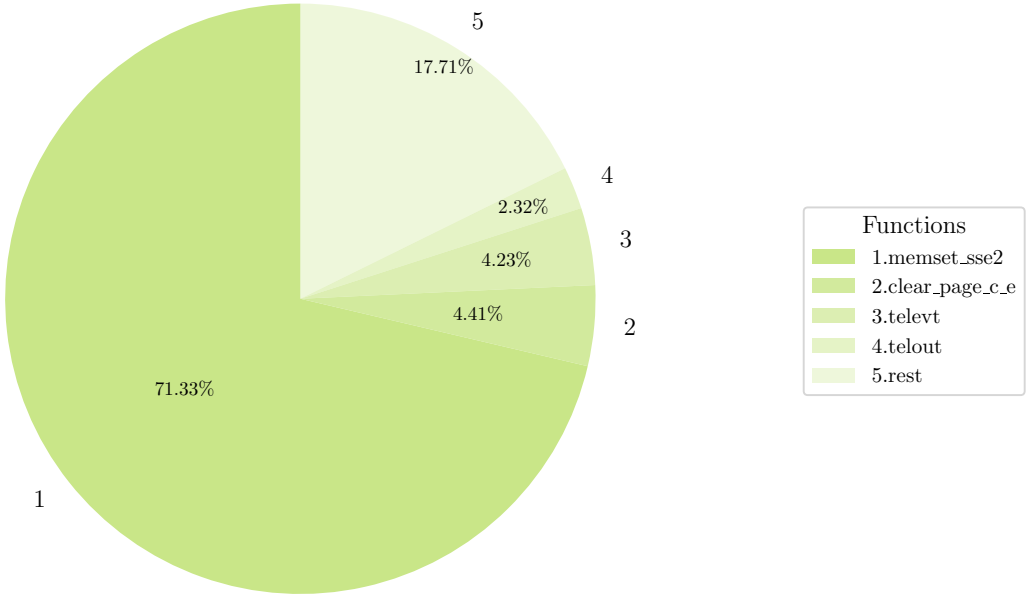

FIGURE 4.5 – Répartition des défauts de cache par fonction dans `CORSIKA`.

CORSIKA présente une accélération de 1.8.

4.7 Validation des résultats

Dans ce travail d'optimisation nous nous sommes fixé l'objectif d'obtenir des résultats identiques à la version originale. Pour cela, nous avons comparé la sortie de la version originale avec celle de la version optimisée. La sortie de CORSIKA consiste en une liste de *bunch* ayant touché les télescopes, avec leur position, direction et temps d'arrivée. Nous avons comparé tous ces paramètres pour chaque *bunch* en exigeant qu'ils soient identiques entre les deux versions.

S'agissant d'un code de simulation Monte-Carlo, quelques précautions ont été nécessaires. En particulier, nous avons fixé la graine du générateur de nombres aléatoires afin d'obtenir des résultats reproductibles d'une exécution à l'autre. En outre, pour prendre en compte la variabilité du développement d'une cascade à une autre, nous avons effectué des comparaisons en simulant jusqu'à 5000 cascades. La comparaison a donc été effectuée sur environ 3.4 milliards de *bunch* produits dans ces simulations sans que nous ayons observé aucune différence entre les deux versions.

Pour des éventuelles optimisations futures qui introduiraient des approximations, des changements algorithmiques ou une réduction de la précision, la validation sera plutôt obtenue sur une base statistique en comparant des distributions de référence.

4.8 Mise en production sur la grille

L'objectif du projet étant d'exécuter la version optimisée sur l'infrastructure de grille actuellement utilisée en production, la portabilité du code représente une contrainte supplémentaire. La stratégie d'optimisation par la vectorisation et en particulier par l'auto-vectorisation visait justement à prendre en compte cette contrainte.

Les centres de calcul de la grille sont équipés de processeurs avec différents modèles de CPU et avec différents niveaux du support des instructions vectorielles. À titre d'exemple, lors d'une campagne de test en juin 2020, nous avons observé la répartition des CPUs en fonction du modèle et des instructions vectorielles supportées tel que présenté sur la Figure 4.6.

Afin de porter la version vectorisée sur la grille, nous avons préparé plusieurs versions du code compilé avec différentes options pour l'activation des différents jeux d'instructions disponibles, i.e. SSE4, AVX, AVX2, AVX-512. Ensuite, nous avons instrumenté les tâches CORSIKA pour qu'au début de leur exécution, elles découvrent quel jeu d'instruction est disponible sur le noeud de calcul et choisissent la version compilée correspondant. De cette manière chaque tâche utilise au mieux les capacités du processeur sur lequel elle s'exécute. Nous avons donc effectué une campagne de test qui a consisté à exécuter sur la grille deux lots de 1000 tâches CORSIKA, l'un avec la version d'origine et l'autre avec la version optimisée. Chaque tâche simulait 50k cascades en plusieurs heures. Le gain mesuré en comparant le CPU global des 2 lots de tâches est de 1.6, compatible avec le gain de 1.8 mesuré sur un serveur dédié avec de l'AVX2. Enfin, aucun problème de portabilité n'a été observé.

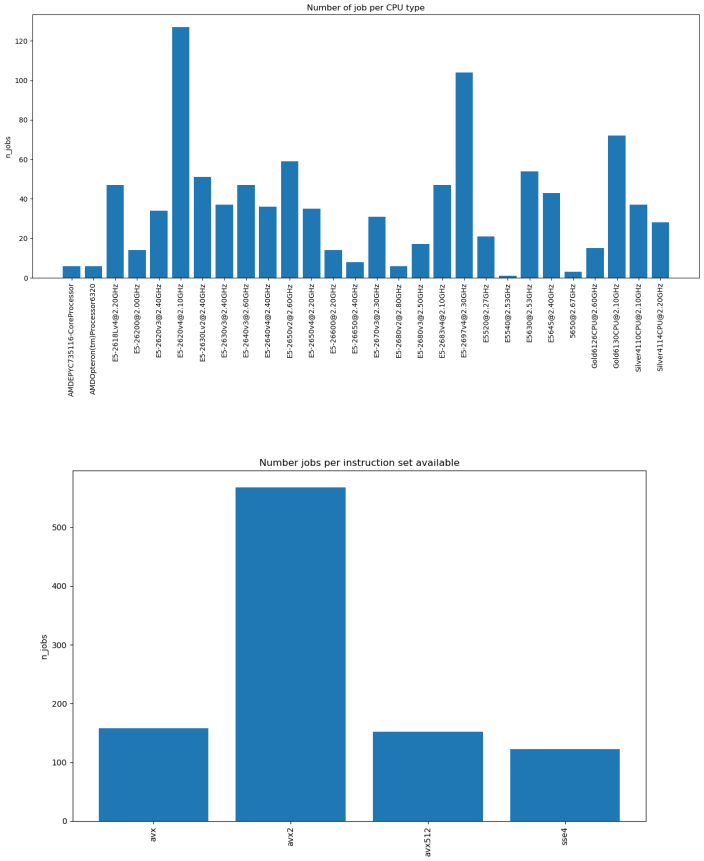

FIGURE 4.6 – Répartition des CPUs en fonction du modèle (en haut) et du jeu d'instructions vectorielles (en bas), telles qu'observées lors d'une campagne de test sur la grille en juin 2020.

À la vue de ces résultats, nous avons utilisé la version optimisée pour accélérer les tâches de simulation de la dernière campagne massive effectuée pendant l'été 2020 (PROD5), cf. Section 3.4. Cette production correspond à environ 434k tâches exécutées sur la grille, pour une consommation CPU globale de 31 millions d'heures HS06. `CORSIKA` représente environ 40% du CPU total, soit 12 millions d'heures HS06. En observant la Figure 4.7, on note que la fraction de tâches exécutées sur des processeurs avec de l'AVX2 et AVX-512 est supérieure à celle observée lors de la campagne de test. En considérant donc d'avoir bénéficié d'une accélération d'au moins 1.6, nous estimons une économie d'environ 7 millions d'heures HS06 de calcul uniquement pour cette campagne.

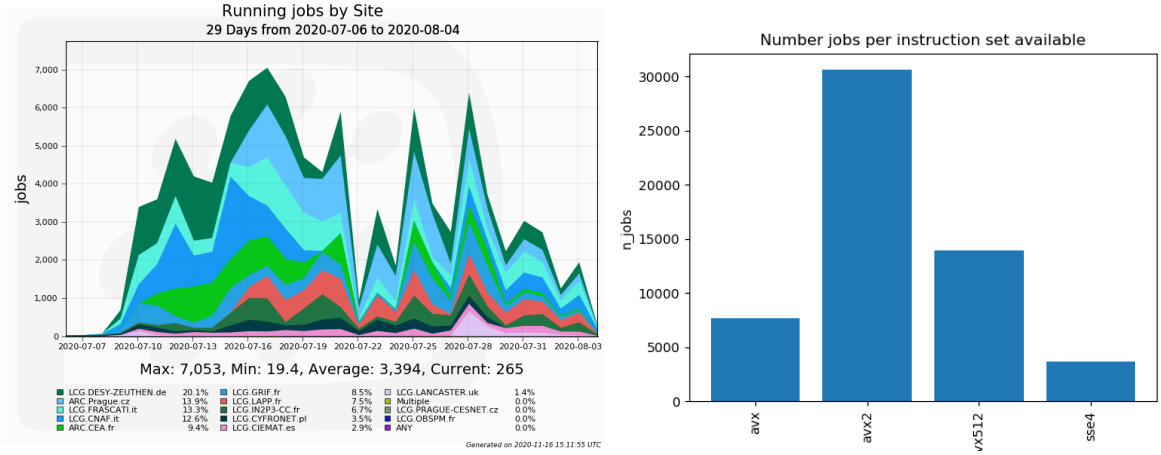

FIGURE 4.7 – À gauche : nombre de tâches concurrentes reparties sur les différents sites, lors de la campagne de simulation PROD5, en juillet 2020. À droite : répartition des tâches en fonction du jeu d'instructions vectorielles observée pendant la PROD5.

4.9 Conclusions et perspectives

La stratégie d'optimisation de `CORSIKA` à l'aide des techniques de vectorisation s'est révélée très efficace. L'étude des accès mémoire a été un deuxième levier indispensable pour améliorer ultérieurement les performances.

Grâce aux optimisations présentées dans ce chapitre, nous avons obtenu une accélération globale égale à 1.8. La réduction des coûts de calcul associés est très importante. Sur environ 100 millions d'heures HS06 par an consommées par les simulations, cela représente une économie d'environ 45 millions d'heures de calcul.

Nous pensons que des optimisations ultérieures demanderaient des changements profonds du code, par exemple au niveau algorithmique. Une autre piste d'optimisation consisterait à réduire la précision de certains calculs, en passant de la double à la simple précision. Cela permettrait de doubler le gain de performances sur les parties de code vectorisées. Cette piste est très intéressante, mais elle n'est pas évidente à mettre en oeuvre. La difficulté principale consiste à cibler les calculs qui peuvent être faits en simple précision sans affecter les résultats finaux au delà des limites acceptables pour les physiciens.

Au lieu de poursuivre ces pistes, nous avons plutôt choisi de contribuer au projet de réécriture de CORSIKA en C++ (projet CORSIKA 8). L'objectif est de profiter de ce nouveau développement pour prendre en compte dès le départ les aspects liés à la performance ainsi que les exigences de CTA, notamment pour le module Cherenkov. Il s'agit par exemple d'étendre la vectorisation au delà de ce qui a été fait dans CORSIKA 7, en exploitant d'autres possibilités de parallélisation.

Un projet de référence sur la vectorisation des codes de transport est le projet GeantV [41]. Ce projet, terminé en fin 2019, a consisté à revoir complètement le framework de Geant4 pour profiter de manière extensive du modèle SIMD en exploitant le parallélisme au niveau des traces. Bien que le gain global obtenu par GeantV a été en dessous des attentes, le projet a produit des recommandations très intéressantes qui nous seront certainement utiles dans le développement de CORSIKA 8. D'ailleurs, dans le cadre du PEPS Astro-Informatique, en décembre 2018 nous avons organisé un atelier [2] au LUPM en invitant des représentants de GeantV, CORSIKA 8 et CTA. Cet atelier a été l'occasion de faire le point sur les stratégies d'optimisation mises en œuvre par les différents projets, et de partager les expériences respectives notamment avec les bibliothèques d'accès aux instructions SIMD et de fonctions mathématiques vectorisées.

2. https://gite.lirmm.fr/cta-optimization-group/cta-optimization-project/-/wikis/lupm_december_2018

Conclusions

Dans ce manuscrit d'habilitation j'ai ainsi présenté deux aspects complémentaires relatifs à l'optimisation de l'utilisation des ressources de calcul pour des applications dans les domaines de la physique des hautes énergies et de l'astrophysique. Au Chapitre 1, j'ai en particulier exposé les défis en termes de calcul et stockage de l'observatoire CTA.

Le premier aspect de l'optimisation que j'ai abordé concerne la gestion des tâches de calcul sur des infrastructures distribuées (*Workload Management System*). Dans le Chapitre 2, j'ai présenté l'implémentation d'un WMS basée sur le mécanisme du *pilot job* par le framework DIRAC, en expliquant les points forts de cette approche. L'un des défis présents et futurs des WMS concerne l'intégration de nouveaux types de ressources, tels que les centres HPC et les clouds privés et commerciaux. Dans la Section 2.7.1, j'ai décrit comment DIRAC permet d'intégrer toutes ces ressources. Cependant, du fait des évolutions régulières des interfaces et des technologies utilisées par les centres de calcul, cet effort d'intégration devra continuer.

En outre, nous assistons de plus en plus à une diversification du matériel : différentes architectures CPU, GPU spécialisées et cartes FPGA. L'utilisation efficace de ce matériel spécifique représente un défi supplémentaire. Afin de bénéficier des capacités de ces ressources spécialisées, les applications nécessitent une connaissance assez détaillée du matériel (architecture CPU, RAM, capacité des différents niveau de cache, modèle de carte GPU, etc.). Les WMS devront donc être capables d'ordonnancer efficacement des tâches en fonction du matériel ciblé. Un axe future de travail consistera donc dans le développement d'un modèle générique permettant une description détaillée de différents types de ressources et une association efficace des tâches aux ressources ciblées.

En outre, à la vue du succès du projet d'application de DIRAC à CTA, j'ai souhaité mettre en évidence les bénéfices pour les communautés de partager des solutions logicielles. Aujourd'hui, cette démarche est considérée comme nécessaire pour réduire les coûts de développement, des opérations et gagner en maintenabilité à long terme [42].

Le deuxième sujet traité dans ce manuscrit concerne l'optimisation de code. Au delà de la pure performance, nous sommes intéressés à la portabilité, à la reproductibilité et à la précision des résultats, qui sont également les thématiques au coeur du projet `Reprises`.

À travers le projet d'optimisation de `CORSIKA`, je me suis intéressée plus spécifiquement aux différents aspects liés à la vectorisation. La vectorisation de la propagation des photons Cherenkov dans `CORSIKA` a permis d'atteindre une amélioration des performances d'un facteur proche de 2, ce qui représente plusieurs dizaines de milliers d'heures de calcul économisées par an. La portabilité de cette solution a été prouvée avec la mise en production sur la grille et son exploitation lors de la dernière campagne à grande échelle de CTA, pendant l'été 2020.

Au delà de `CORSIKA`, de nombreux codes en physique pourraient bénéficier de la vectori-

sation, notamment lorsqu'il s'agit de manipuler de façon régulière des images, des vecteurs, des matrices ou des tenseurs. De mon point de vue, la vectorisation est l'une des approches les plus prometteuses dans la recherche de performances pour des applications en physique et en astrophysique et elle devrait être appliquée de manière beaucoup plus systématique. Pour les nouveaux développements, une réflexion devrait être menée en amont sur les conditions nécessaires à la vectorisation, comme le choix du format de données, les choix algorithmiques et les parallélismes à exploiter.

À travers ce projet, j'ai souhaité également souligner l'intérêt des collaborations entre physiciens et informaticiens pour les deux communautés. L'intérêt pour la communauté de physique est assez immédiat, car elle bénéficie de l'expertise et des outils développés dans le cadre de la recherche en informatique. Dans la Section 4.5, j'ai mentionné par exemple le gain significatif qu'on obtient en utilisant la bibliothèque de fonctions élémentaires vectorisées 'vector libm', issue du projet MetaLibm [3] sur la génération automatique de fonctions élémentaires.

De l'autre côté, les codes de physique, par leur complexité et par les contraintes de performances et portabilité associées, représentent d'excellents cas d'applications pour la communauté informatique. En effet, la résolution de problèmes complexes, requière souvent le développement de nouvelles méthodes ou outils au delà de l'état de l'art. En outre, les outils et méthodologies développées pour résoudre un problème spécifique peuvent, dans certains cas, être généralisées pour un grand nombre d'applications.

Dans le cadre de l'optimisation de `CORSIKA`, nous avons étudié les limitations de l'optimisation automatique et appliqué des transformations manuelles du code afin d'activer l'auto-vectorisation. En partant de cette étude, nous envisageons d'extraire une méthodologie de transformation de code pour l'activation de l'auto-vectorisation qui pourrait être généralisée et intégrée dans un outil d'optimisation automatique.

Une deuxième perspective concerne l'exploration du calcul multi-précision pour des applications en physique. Ceci est également l'un des thèmes de `Reprises` et dans un contexte plus général, l'un des sujets de recherche de l'équipe DALI du LIRMM. En effet, l'approche des physiciens pour obtenir des résultats les plus précis possible, est d'avoir recours systématiquement au calcul en double précision. Or, il peut être pertinent dans certains cas de réduire la précision afin d'accélérer les calculs, car tous les calculs n'ont pas besoin du même niveau de précision. Pour des applications vectorisées, le passage de la double à la simple précision permet de gagner immédiatement un facteur 2. En outre, certains matériels récents, comme les FPGA, ne possèdent pas de support natif de la double précision, et ne peuvent émuler celle-ci qu'à un coût prohibitif. D'autres, comme les GPUs, la pratiquent nativement mais avec un surcoût très important. À l'inverse, le matériel support de précisions réduites se développe, poussé par l'expansion rapide de l'apprentissage profond. En conclusion, l'applicabilité du calcul multi-précision aux codes de physique est une question ouverte et l'un des sujets auxquels la communauté de physique a tout intérêt à contribuer en synergie avec les informaticiens.

3. `http://www.metalibm.org/ANRMetaLibm/`

Appendices

Annexe A

Contrats de Recherche

A.1 Le projet DIRAC@IN2P3

Le projet DIRAC@IN2P3 est né de l'étroite collaboration existante entre A. Tsaregorodtsev (Ingénieur de recherche au CPPM et porteur du projet), J. Bregeon (Chargé de recherche au LUPM et actuellement au LPSC) et moi même, dans le cadre du consortium international DIRAC. L'objectif du projet est de promouvoir le développement de DIRAC autour de 3 axes principaux décrits dans la Section A.1.2. Ceci, dans une optique de généralisation du code pour qu'il soit utilisable par différentes communautés. L'un des objectifs initiaux était également de structurer les contributions françaises autour de DIRAC en rassemblant les développeurs, les communautés d'utilisateurs et les opérateurs du service DIRAC de France Grilles (FG-DIRAC) [1]. Le projet a été proposé fin 2016 au Directeur Adjoint Scientifique 'Calcul et données' de l'IN2P3 à l'initiative du LUPM et du CPPM. La présentation du projet, tel que proposé à son lancement est exposée dans les sections suivantes. Le projet a été accepté et a démarré officiellement en 2017 avec un financement annuel variable entre 6000 et 10000 euros en fonction des années. Les partenaires au lancement du projet étaient : CPPM, LUPM, CC-IN2P3 avec également la participation de 2 partenaires hors IN2P3, i.e. l'Université de Bordeaux et le laboratoire CREATIS (INSERM). En 2019, le IPHC et le LPSC ont également rejoint le projet, qui compte actuellement 13 membres et 2.8 ETP.

A.1.1 Objectifs scientifiques

L'objectif du projet DIRAC présenté dans ce document est le développement de nouvelles composantes et fonctionnalités du logiciel DIRAC (`http://diracgrid.org/`). DIRAC est un logiciel libre (sous licence GPL V3) de type *interware* pour gérer des ressources de calcul distribué. Il est utilisé par de nombreuses expériences dans lesquelles l'IN2P3 est impliqué dont CTA, ILC, Belle II, BES et LHCb au CERN. L'expérience LHCb est à l'origine du projet et en reste l'utilisateur principal. Depuis 2014, le projet s'est constitué en Consortium, associant 3 partenaires fondateurs : le CNRS au travers du CPPM, le CERN et l'Université de Barcelone. L'objectif du Consortium DIRAC est de formaliser la collaboration entre les partenaires pour assurer le développement et la maintenance de DIRAC (*Agreement for De-*

1. `http://www.france-grilles.fr/catalogue-de-services/fg-dirac/`

velopment and Maintenance of the DIRAC Software for Distributed Computing). Récemment d'autres partenaires ont rejoint le Consortium, dont KEK, IHEP et le LUPM/Université de Montpellier en cours de signature. Ce modèle a permis d'impliquer de nouveaux développeurs et d'explorer de nouvelles idées. Cette synergie est très avantageuse pour les différentes expériences qui profitent des développements faits en dehors de leur collaboration. Dans ce contexte, les objectifs du projet consistent d'une part à contribuer au cœur du logiciel au travers de la modernisation des outils et de l'architecture et d'autre part à jouer un rôle de leadership dans les domaines suivants : gestion des *workflows* ; gestion de données en masse ; gestion des ressources de calcul et de stockage.

A.1.2 Plan de développement

Le plan proposé est basé sur 3 axes de développement principaux :

- Gestion des workflows
- Développement d'une nouvelle composante générique - Production System - sur un modèle data–driven
- Gestion des ressources de calcul et de stockage
  - Intégration des ressources de type HPC et *cloud* (e.g. méta–ordonnanceur pour la réservation intelligente et efficace des machines virtuelles sur des clouds de calcul)
  - Intégration des ressources de stockage cloud (dépôts d'objets) et des dépôts de données en forme de base de données
- Gestion de données en masse
  - Développement de nouvelles fonctionnalités pour supporter le mode de travail *Big Data*

Les deux premiers axes étant les axes prioritaires pour une durée prévue du projet de 24 mois. En plus de ces axes de développement principaux, le projet vise à améliorer les performances du logiciel en introduisant des technologies modernes, e.g. systèmes de *Message Queuing* et bases de données non relationnelles. Les tests seront réalisés sur une instance dédiée dans le cloud du LUPM. Enfin, pour le processus de validation et de vérification, le projet s'appuiera sur des cas concrets d'utilisation, fournis par CTA et LHCb, en utilisant les instances DIRAC de CTA et éventuellement de France Grilles.

A.1.3 Résultats attendus

Les bénéfices attendus pour les expériences utilisatrices de DIRAC sont multiples :

- Simulations et traitement des données plus stables et efficaces
- Moins d'ETP nécessaires pour le suivi des productions
- Temps réduit entre acquisition des données (ou production des simulations) et distribution des produits scientifiques
- Gestion des données en masse plus efficace
- Accès simplifié et efficace à de nouveaux types de ressources de calcul et de stockage (HPC, cloud)

A.1.4 Succès et perspectives

Le bilan du projet à 4 ans de sa création est très satisfaisant autant du point de vue des réalisations logicielles que du point de vue de la production scientifique. En outre, bien que les contributeurs principaux restent le CPPM et le LUPM, le projet a su créer une synergie très fructueuse entre tous les partenaires. Nous tenons des réunions mensuelles durant lesquelles nous faisons le point sur l'état de DIRAC dans son ensemble, nous discutons des différents projets individuels, des éventuelles difficultés rencontrées et des possibles solutions. Deux fois par an, nous tenons également une réunion face-à-face, qui est l'occasion d'approfondir des sujets spécifiques et de planifier les actions futures. Ces échanges s'avèrent très bénéfiques pour l'ensemble du groupe, ce qui contribue au succès du projet.

Parmi les principales réalisations en 2019/2020, on mentionne :

- *Production System* pour la gestion de *workflows* inclus dans la distribution officielle de DIRAC (L. Arrabito, LUPM)
- Prototype d'un système d'authentification/autorisation basé sur la technologie OAuth/OIDC pour les services DIRAC (A. Lytovchenko, CPPM)
- Exploration de l'application du language Go pour le protocole de communication avec des applications distribuées (C. Meessen, CPPM)
- Participation au projet ESCAPE (via le CPPM et CTA) - *cluster of Astronomy Particle physics ESFRI research infrastructures*
  - DIRAC dans le catalogue des logiciels et des services d'ESCAPE
  - Développement du portail d'application CORSIKA dans le cadre de DIRAC (L. Fusco, CPPM)
- Intégration de ressources cloud
  - Démonstrations dans le cadre de CTA, France Grilles-DIRAC et DIRAC4EGI
  - Développement d'interfaces de connexion directes pour Openstack et OpenNebula
- Intégration de ressources HPC
  - Connexion du mésocentre AMU et du centre HPC Govorun à Dubna
  - Intégration de ressources GPU (CREATIS)

Les membres du projet présentent également leurs travaux de manière régulière à plusieurs conférences internationales. Ci-dessous la liste des *proceedings* pour la période 2019-2020 :

- L. Arrabito et al., *The Cherenkov Telescope Array production system for data-processing and Monte Carlo simulation*, EPJ Web of Conferences 214, 03052 (2019)
- S. Pop et al., *Exploiting GPUs on distributed infrastructures for medical imaging applications with VIP and DIRAC*. In *42nd international convention on information and communication technology, electronics and microelectronics*, pages 190–195, Opatija, Croatia, May 2019
- L. Arrabito et al., *A DIRAC-based prototype for the Cherenkov Telescope Array data management, processing and simulations*, *Proceedings at the 29th Astronomical Data Analysis Software and Systems (ADASS)*, 2019

- V. Korenkov et al., *Integration of the JINR Hybrid Computing Resources with the DIRAC Interware for Data Intensive Applications*, *Proceedings of International Conference on Data Analytics and Management in Data Intensive Domains, DAMDID*, 2019

Au niveau national, nous participons régulièrement à différentes journées thématiques :

- Journées Calcul et données 2019
  - S. Pop et al., *Exploiting GPUs for medical imaging applications with VIP and Dirac*
- Journées Informatiques de l'IN2P3-IRFU 2018
  - L. Arrabito et al., *Projet DIRAC@IN2P3*
- Journées SUCCES 2017
  - L. Arrabito et al., *Le projet DIRAC à l'IN2P3*
  - L. Arrabito : Participation à la table ronde 'Usages du cloud et cloud fédéré pour les utilisateurs'

Depuis sa création, la participation au projet n'a cessé d'augmenter à la fois en termes de laboratoires partenaires, qu'en termes de nombre de personnes impliquées. Ceci démontre l'intérêt croissant autour de DIRAC, autant de la part des communautés d'utilisateurs que de la part des fournisseurs du service (FG-DIRAC). Il est à noter que le projet a également bénéficié de ressources supplémentaires grâce à la participation du CPPM à plusieurs projets européens (EGI-Inspire, EOSC-Hub, ESCAPE).

Dans cet élan, le projet compte continuer à développer le logiciel et le service dans les années à venir. Du point de vue logiciel, les actions prévues en 2021 sont les suivantes :

- Mise en production du système d'authentification/autorisation de DIRAC (OAuth/OIDC)
- Intégration des centres HPC Lobachevsky et Nizhni Novgorod
- Prototype de services DIRAC en langage Go
- Dans le cadre d'ESCAPE : mise en production du portail CORSIKA connecté à DIRAC

Du point de vue du service, dans le cadre du futur projet européen EGI ACE, il est prévu de fusionner les services DIRAC opérés par France Grilles (FG-DIRAC) et EGI (DIRAC4EGI). L'objectif étant de rationaliser les coûts des opérations en maintenant une seule instance qui servirait toutes les communautés.

A.2 CTAOptSim, CNRS PEPS Astro-Informatique

En 2017, j'étais à l'origine d'une collaboration avec l'équipe DALI du LIRMM autour d'un projet d'optimisation du logiciel de simulation de cascades atmosphériques `CORSIKA`. Les résultats préliminaires de ce projet étant encourageants, nous avons décidé de répondre l'appel à projet PEPS [2] Astro-Informatique de la Mission pour les initiatives transverses et interdisciplinaires du CNRS. La proposition a été acceptée et financée pendant deux ans (2018-2019) à hauteur de 7000 euros par an. Le projet est présenté plus en détail dans les sections suivantes.

2. `https://www.cnrs.fr/mi/spip.php?article8`

A.2.1 Contexte

Le projet CTA (Cherenkov Telescope Array) [1] a pour objectif la construction et l'exploitation d'un réseau d'environ une centaine de télescopes Cherenkov sur les sites de La Palma (Canaries, Espagne) et Paranal (ESO, Chili). CTA ouvrira une nouvelle ère dans l'astronomie gamma de très haute énergie grâce à des performances biens supérieures à celles des expériences en cours. Le consortium en charge de la conception et de la construction de CTA regroupe environ 1200 membres venant d'une centaine d'instituts de recherche de 25 pays : la France est l'un des principaux acteurs du projet via les contributions du CNRS et du CEA. Les simulations Monte Carlo (MC) jouent un rôle majeur au coeur du projet. Les simulations ont tout d'abord permis de définir le design global de CTA (nombre et dimension des télescopes), puis ont permis récemment de guider certains choix stratégiques comme la sélection des meilleurs sites d'implantation et la définition de la géométrie optimale des réseaux [34] [35]. Ces productions, gérées par le LUPM depuis 2013, utilisent typiquement 8000 cœurs en simultané distribués sur une vingtaine de sites de la grille européenne EGI. En phase d'opération, les simulations MC seront nécessaires afin de calculer pour chaque intervalle de temps donné, les fonctions de réponse des télescopes à la lumière Cherenkov émise par les gerbes atmosphériques. C'est à travers ces fonctions de réponse que les astrophysiciens pourront découvrir les sources astronomiques et en déterminer les caractéristiques spatiales, spectrales et temporelles. Le temps de calcul associé aux simulations est et restera très important, i.e. environ 200 millions d'heures CPU normalisées par an, selon les estimations actuelles.

A.2.2 Objectifs scientifiques

Les simulations MC de CTA sont composées de deux étapes réalisées par deux logiciels dédiés. La première consiste en la simulation détaillée des gerbes induites par les rayons cosmiques de haute énergie, et leur développement dans l'atmosphère. Pour cette partie, CTA utilise le logiciel `CORSIKA`, un logiciel de référence dans la communauté étudiant les rayons cosmiques. La deuxième étape consiste à simuler la réponse des télescopes à la lumière Cherenkov (logiciel `sim_telarray`). Selon nos estimations, `CORSIKA` consomme environ 70% du CPU global dans la chaîne de simulation. Compte tenu du temps de calcul global dédié aux simulations MC dans CTA, une amélioration de l'efficacité du calcul permettrait, d'un point de vue projet, d'envisager une réduction des infrastructures et donc des coûts à volume de simulation constant. En outre, du point de vue scientifique, on s'offre différentes possibilités : produire des fonctions de réponse dans un délai plus court (intéressant pour la science des évènements transitoires, cf. GW 170817 [43]), ou réduire les incertitudes statistiques et systématiques sur chaque fonction de réponse en augmentant la statistique des évènements simulés et en élargissant l'espace des paramètres considérés ainsi que leur granularité. Les simulations Monte Carlo étant toutes indépendantes et très nombreuses, ce processus de simulation est, par nature, massivement parallèle (cœurs indépendants) et il opère sur de très grands jeux de données. Il est donc inutile d'avoir recours à la parallélisation d'une simulation pour réduire son temps de calcul. En effet, il sera plus efficace et plus simple d'exploiter des ressources supplémentaires en réalisant de nouvelles simulations ou en distribuant les simulations existantes sur plus de cœurs plutôt que de paralléliser l'application. Aujourd'hui,

les différentes générations de processeur proposent des améliorations architecturales avec des adaptations du jeux d'instructions comme par exemple les instructions vectorielles (e.g. AVX) qui permettent des gains importants en temps d'exécution (x2 à x16) et en consommation d'énergie. De plus, les micro-architectures de processeur possèdent des caractéristiques différentes comme par exemple les hiérarchies mémoires. Qu'elles soient de nature architecturales ou micro-architecturales, les programmes doivent prendre en compte toutes ces spécificités matérielles afin d'en exploiter pleinement les performances. Or le logiciel `CORSIKA`, écrit en Fortran avec certaines bibliothèques écrites en C, est écrit sans aucune optimisation particulière afin de pouvoir tourner sur une architecture de processeur généraliste. L'optimisation du logiciel `CORSIKA` apparaît comme une nécessité. La mise en production du logiciel se fera sur des « fermes » de calcul avec des architectures différentes. Il faudra donc dans un second temps mettre en place un processus de mise au point automatique des optimisations en fonction de l'architecture et de la micro-architecture cible. L'objectif scientifique des informaticiens se situe précisément dans l'automatisation du processus de mise au point. Le point de départ est le développement d'une version de référence qui sera optimisée via l'utilisation de logiciels existants représentant l'état de l'art (e.g. Intel Advisor) et ce pour une ou plusieurs architectures cibles. Elle pourra être très rapidement intégrée à la chaine de calcul pour définir les gains effectifs de performance. Le premier objectif est d'étudier l'écart de performance entre une application optimisée automatiquement par des outils relevant de l'état de l'art actuel et une optimisation manuelle. Nous pensons que l'optimisation manuelle sera plus efficace que les solutions automatiques de l'état de l'art, notamment de part l'ancienneté et la complexité du code `CORSIKA`. Le second objectif est donc d'identifier, dans le cadre des applications visées par le projet, les limitations des solutions automatiques, puis de proposer des solutions pour améliorer l'approche automatique et les implanter. Une des originalités de l'approche menée est d'intégrer une optimisation bi-critères (performance et précision numérique) particulièrement bien adaptée à l'objectif de vectorisation de ces développements. Le projet `CTAOptSim` permet ainsi de contribuer aux recherches actuelles dans le domaine de l'optimisation automatique des performances de codes de calcul numérique et de l'équilibre précision-performance, questions importantes pour la communauté de l'informatique HPC, et en même temps d'obtenir des versions d'exploitation de traitement CTA qui profiteront de gains de performance significatifs. Les objectifs scientifiques des astrophysiciens et des informaticiens s'intersectent donc pleinement.

A.2.3 Présentation des partenaires

Les partenaires de `CTAOptSim` regroupent toutes les compétences nécessaires à la réussite du projet. Le LUPM (UMR 5299) est aujourd'hui fortement impliqué dans CTA, que ce soit au niveau de la science, de l'instrumentation et de l'informatique. J. Bregeon (CR, LUPM), astrophysicien et porteur du projet, fait partie du groupe d'experts en charge des simulations et analyses de CTA. Ses connaissances du logiciel `CORSIKA` et de la physique associée sont d'une grande aide pour le travail des informaticiens. En outre, il est responsable de l'intégrité des résultats obtenus avec le code optimisé. L. Arrabito (IR, LUPM), ingénieur en informatique et physicienne de formation, est responsable depuis cinq ans des productions MC sur la grille de calcul EGI, via le développement d'un système de production dédié [26]. Grâce à ses compétences en génie logiciel et son expérience dans un environnement de pro-

duction à grande échelle, elle assure la qualité des développements et leur maintenabilité à long terme. Les recherches menées par l'équipe DALI du LIRMM (UMR 5506) à l'Université de Perpignan contribuent à l'amélioration de la performance et de la qualité numérique du logiciel scientifique. Ph. Langlois (PU, DALI) est un spécialiste du logiciel numérique et de l'arithmétique des ordinateurs, en particulier pour les aspects de validation et d'optimisation numérique des calculs en arithmétique flottante. D. Parello (MCF, DALI) est spécialisé dans la compilation et l'optimisation de codes, avec une orientation récente vers les traitements à forte composante numérique. Les recherches de G. Revy (MCF, DALI) concernent la génération et la validation automatiques de codes en virgule fixe et virgule flottante, en particulier d'opérateurs arithmétiques, et actuellement l'ajustement automatique précision-performance.

A.2.4 Planning

Le projet `CTAOptSim` est composé de trois phases. La première phase consiste en l'identification des opportunités d'optimisation de `CORSIKA` en incluant également l'approche par réduction de la précision numérique. Elle devrait s'étaler sur le premier trimestre 2018. La deuxième phase consiste en l'implantation d'une première version optimisée de référence pour une architecture cible et en l'évaluation des gains effectifs de performance. Elle sera réalisée dans le cadre d'un stage de M2 (février à juillet 2018). Les cinq derniers mois du projet seront consacrés d'une part à la mise en production et à la validation de la version optimisée de référence et d'autre part à l'implantation d'un deuxième niveau d'optimisation utilisant l'approche par réduction de la précision. Enfin, une fois les problèmes compris nous entamerons l'étude de nouvelles solutions pour l'optimisation automatique.

A.2.5 Résultats attendus et perspectives

Comme nous l'avons évoqué, le premier résultat attendu du projet `CTAOptSim` est le développement d'une version optimisée du logiciel `CORSIKA` avec pour objectif sa mise en production dans le cadre des simulations de CTA sur des fermes de calcul ayant des architectures hétérogènes, et donne un cadre, un cas concret, pour le développement d'outils de mise au point automatique de code optimisé en fonction de l'architecture et de la micro-architecture cible. À plus long terme, nous souhaitons aussi faire perdurer cette collaboration interdisciplinaire entre le LUPM et DALI, afin d'atteindre des objectifs scientifiques plus ambitieux, que ce soit dans le cadre de CTA ou en s'appuyant sur des nouveaux cas d'étude issus des thématiques de recherche en astrophysique du LUPM.

A.2.6 Succès et perspectives

Grâce à une implication forte des deux équipes, le projet a obtenu d'excellents résultats, notamment avec le développement de la version de `CORSIKA` optimisée qui est actuellement utilisée en production par le consortium CTA. L'aspect interdisciplinaire du projet, physiciens et informaticiens travaillant de concert, représente clairement une vraie plus-value pour le projet. Le LUPM a apporté la connaissance du fonctionnement de `CORSIKA` et en particulier du cas d'application de CTA. Les informaticiens de l'équipe DALI, quant à eux, ont apporté toute leurs expertise dans plusieurs domaines, i.e. optimisation de code, processus

de compilation, vectorisation, développement de fonctions élémentaires et précision numérique. Personnellement, j'ai eu l'opportunité de collaborer de manière très étroite avec les différents membres de l'équipe DALI et de m'initier à tous ces aspects. Le travail commun a permis d'établir une méthodologie adaptée portant à une optimisation significative d'une partie importante du code en très peu de temps.

Dès la première année du projet, nous avons obtenu des résultats très intéressants. Tout d'abord, nous avons étudié le comportement du programme à l'aide de différents outils de profilage. Ensuite, en concertation avec Konrad Bernlöhr (MPIK Heidelberg), développeur du module `iact/atmo`, nous avons procédé à l'optimisation des algorithmes d'interpolation des profils d'atmosphère. Un premier travail de vectorisation de la propagation des photons Cherenkov de `CORSIKA` a été également réalisé, ce qui nous a permis d'obtenir une accélération de 1.2. Par cette occasion, nous avons également mené une étude comparative de différentes bibliothèques de vectorisation et nous avons appliqué avec succès la bibliothèque de fonctions élémentaires vectorisée 'vector libm' [40].

Fin 2018, nous avons organisé un atelier élargi dédié au thème de l'optimisation des codes de simulation Monte Carlo à Montpellier dans les locaux du LUPM [3]. Étaient présents, outre les participants au projet `CTAOptSim`, deux représentants du groupe *Analysis and Simulation Working Group* de CTA, un représentant du projet `GeantV` (version vectorisée du code de simulation Monte-Carlo `Geant4`), deux développeurs de `CORSIKA` ainsi que plusieurs chercheurs, postdocs et doctorants du laboratoire. Cet atelier a été l'occasion de faire le point sur les stratégies d'optimisation mises en œuvre par les différents projets et d'établir les axes de travail pour la suite du projet. Un point saillant des discussions a été une convergence commune sur la nécessité de traiter la précision des calculs (e.g. float vs double) avec des outils adaptés pour pousser plus en avant l'utilisation des instructions vectorielles. Nous avons décrit plus en détail les travaux réalisés pendant cette première phase du projet dans les *proceedings* de la conférence CHEP, qui s'est tenue à Sofia en 2018.

Au cours de la deuxième année, en particulier dans le cadre du stage de Master d'Adnane Khattabi, nous avons étudié de manière plus précise les limitations de la vectorisation automatique et restructuré le code pour dépasser ces limitations. Cette restructuration nous a permis d'étendre d'avantage la vectorisation du module Cherenkov et d'améliorer ultérieurement les performances en atteignant une accélération globale de 1.5. Nous avons présenté régulièrement les avancées du projet devant le consortium CTA, ainsi que dans les ateliers annuels du projet `CORSIKA 8` et au sein du projet `Reprises`. Ces travaux ont fait également l'objet d'une publication dans le journal *Computing and Software for Big Science* [37].

Vu le succès du projet `CTAOptSim` et des liens forts qui se sont instaurés entre le LUPM et l'équipe DALI/LIRMM, nous avons décidé de poursuivre le projet dans le cadre d'une thèse co-dirigée entre les deux équipes (cf. Annexe B.2).

A.3 Le projet REPRISES

Le projet `Reprises` a été proposé fin 2017 au Directeur Adjoint Scientifique 'Calcul et données' de l'IN2P3 à l'initiative de plusieurs laboratoires : LAL, LUPM, LLR, LPC, IPHC,

3. `https://gite.lirmm.fr/cta-optimization-group/cta-optimization-project/-/wikis/lupm_december_2018`

LAPP et IPNO.

Le projet se propose d'étudier les problématiques de performance, portabilité et reproductibilité pour les codes de physique développés au sein de l'institut. L'objectif est d'aider l'institut à s'orienter dans le maquis des nouveaux matériels de calcul, méthodes et outils logiciels pour gagner en performance sans trop sacrifier en portabilité, lisibilité et précision. Le projet a été rapidement accepté et a pu démarrer officiellement en fin 2017 avec un financement annuel à hauteur de 7000 euros. Au niveau international, le projet `Reprises` s'inscrit dans le cadre des activités de la *HEP Software Foundation*[4].

Pour ma part, j'ai rejoint le projet `Reprises` dès sa création, en lien avec mes activités sur l'optimisation du logiciel `CORSIKA`, et depuis le printemps 2020, j'ai repris la coordination du projet.

A.3.1 Objectifs scientifiques

Les nouvelles architectures matérielles posent un défi majeur aux applications de nos communautés, dont la durée de vie est très longue (10 à 20 ans typiquement) alors que chaque génération de matériel (tous les 2 ans) apporte de nouvelles spécificités qu'il faut utiliser pour bénéficier des améliorations de performance. Plus particulièrement dans notre institut, l'utilisation des grilles de calcul, assemblage de matériel hétéroclite (par opposition aux supercalculateurs), constitue un défi supplémentaire pour la portabilité du code, de ses performances, et pour la reproduction des résultats. La recherche de reproductibilité est le fil rouge de ce projet, tant en termes de précision des résultats numériques, qu'en termes de vitesse de calcul. Côté précision, de nouveaux outils d'arithmétique stochastique promettent de mieux maitriser nos erreurs de calcul flottant, en vue d'éviter des calculs inutilement précis, ou d'identifier des calculs instables nécessitant une plus grande précision. Côté vitesse, pour qu'un code puisse mieux résister à un changement de matériel, nous voulons explorer toutes les stratégies disponibles telles qu'OpenCL, la génération de code, les langages dédiés, les directives portables, les outils de remaniement de code. Toutes les expériences de notre institut sont concernées. Parmi les chercheurs et ingénieurs qui se sont déjà manifestés, on retrouve des acteurs de AMS, ATLAS, BELLE, CMS, CTA, FERMI, GEANT4 GATE, LHCB, LSST...

A.3.2 Plan de développement

Il nous apparaît essentiel de mener ce projet en étroite collaboration avec des laboratoires d'informatique. Des contacts existent déjà avec Telecom Sud-Paris (Gael Thomas), le LRI (Joel Falcou), l'École Centrale de Nantes (Richard Randriatoamanana, Hugues Digonnet), EDF (François Fevotte), le LIRMM (Philippe Langlois, David Parello, Guillaume Revy). Pour leur permettre d'intervenir concrètement, nous devrons sélectionner quelques applications ou mini-applications pilotes de notre discipline, et les reconditionner afin d'être facilement utilisables par des tiers. Par ailleurs, chaque partenaire IN2P3 rejoint le projet avec l'envie d'explorer certaines technologies particulières en vue d'améliorer le logiciel de nos expériences. Chacun devra faire en sorte de partager son savoir-faire en la matière, afin

4. `https://hepsoftwarefoundation.org/`

que nous puissions monter ensemble en compétence, et rechercher un consensus sur les technologies les plus prometteuses. Il faudra également s'accorder sur les outils et les procédures permettant d'évaluer les alternatives de façon cohérente. Quand applications, technologies et procédures seront clarifiées, le gros du travail consistera à porter les applications, à les exécuter sur différents matériels, et à analyser les résultats. Des aller-retour sont à prévoir avec nos collaborateurs des laboratoires d'informatique. Suivra une étape de dissémination des résultats : tutoriels, publications, contributions au logiciel officiel des expériences. La durée du projet a été fixée à 36 mois. Une charte est en cours de rédaction avec le Labex P2IO pour accéder au matériel acquis dans le cadre de ses projets GridCL et ACP, et l'IAS, laboratoire non IN2P3 du Labex, a manifesté son souhait d'être associé au projet Reprises.

A.3.3 Résultats attendus

Le portage des applications pilotes devrait permettre de produire quelques démonstrateurs. Nous planifions au minimum un prototype capable de tourner à la fois sur GPU et sur Xeon Phi, et un prototype capable de se vectoriser sur des processeurs différents. Nous aimerions également, sur au moins une application, démontrer le gain d'une précision réduite, et/ou d'une précision augmentée à bon escient. Les technologies les plus prometteuses feront l'objet de tutoriels en ligne, si possible en s'appuyant sur des conteneurs, afin que tous les ingénieurs et chercheurs de l'IN2P3 puissent prendre conscience des enjeux et essayer les outils et bibliothèques qui permettent de mieux profiler du code moderne (donc parallèle) et d'améliorer la reproductibilité des performances et des résultats. L'ambition essentielle de ce projet, c'est d'aider notre communauté dans son ensemble à sortir du modèle obsolète du « programme séquentiel à base de double ».

A.3.4 Succès et perspectives

Ces trois premières années du projet ont été extrêmement productives. Nous tenons des réunions mensuelles durant lesquelles nous discutons de nos projets respectifs et partageons les problèmes ou solutions spécifiques rencontrées. Ces échanges s'avèrent extrêmement bénéfiques pour tous. Nous tenons de plus deux réunions en face-à-face par an pour approfondir certains points et planifier nos actions futures.

Au delà des études individuelles des membres du groupe, nous menons également des actions collectives de formation et dissémination. En 2019, dans le cadre de l'exercice de prospective nationale de l'IN2P3 [5] pour les dix années à venir (2020-2030), le projet `Reprises` a contribué au groupe thématique 'Calcul, Algorithmes et Données' avec une note technique très détaillée sur les aspects de performance, portabilité et reproductibilité du calcul [6]. Afin de diffuser ce travail au sein de l'institut, en 2020 nous avons entamé la rédaction d'un guide logiciel de performance durable sous la forme d'un site web à destination des chercheurs. Des formations autour du calcul performant, reproductible et précis seront également organisées dès que la situation sanitaire le permettra à nouveau.

5. `https://prospectives2020.in2p3.fr/`

6. `https://codeursintensifs.pages.in2p3.fr/Reprises/docs/prospective-gt09-reprises-rapport-v3.pdf`

Enfin, le projet est également particulièrement productif en termes de publications et présentations aux conférences. Ci-dessous la liste des publications les plus emblématiques :

- Revues :
  - L. Arrabito et al, Optimizing Cherenkov photons generation and propagation in CORSIKA for CTA Monte-Carlo simulations, *Computing and Software for Big Science* (2020)
  - A. Beck et al., Adaptive SIMD optimizations in particle-in-cell codes with fine-grain particle sorting, *Computer Physics Communications*, Volume 244, (2019)
  - P. Aubert et al, Polynomial data compression for large-scale physics experiments, *Computing and Software for Big Science* (2018)
- Conférences :
  - G. Grasseau et al, A deep neural network method for analyzing the CMS High Granularity Calorimeter (HGCAL) events (*proceedings* CHEP 2019)
  - L. Arrabito et al, Performance optimization of the air shower simulation program for the Cherenkov Telescope Array (*proceedings* CHEP 2018)
  - H. Grasland et al, Floating-point profiling of ACTS using Verrou (proceedings CHEP 2018)
  - G. Grasseau et al, Deployment of a Matrix Element Method code for the ttH channel analysis on GPU's platform (*proceedings* CHEP 2018)
  - P. Aubert et al, Data Analysis with SVD for Physical Experiments, 18th SIAM Conference on Parallel Processing for Scientific Computing (*proceedings* 2018)

Annexe B

Encadrements

B.1 Stage de Master

Pendant la deuxième année du PEPS Astro-Informatique `CTAOptSim`, j'ai co-encadré avec David Parello, le stage d'Adnane Khattabi, dont le sujet est décrit ci-dessous. Adnane était un élève ingénieur en troisième année à l'Enseirb-Matmeca de Bordeaux en option PRCD (parallélisme, régulation et calcul distribué). Le stage s'est déroulé au printemps 2019 et a permis d'obtenir d'excellents résultats (voir Section 4.5), notamment le développement d'une version optimisée de `CORSIKA` très proche de la version finale actuellement en production. Adnane a présenté ses résultats devant le groupe de travail *Analysis and Simulation Working Group* de CTA et contribué à la rédaction de la publication dans le journal *Computing and Software for Big Science* :

- Luisa Arrabito, Konrad Bernlöhr, Johan Bregeon, Matthieu Carrère, Adnane Khattabi, Philippe Langlois, David Parello, and Guillaume Revy. Optimizing Cherenkov Photons Generation and Propagation in CORSIKA for CTA Monte-Carlo Simulations. *Comput. Softw. Big Sci.*, 4(1) :9, 2020.

B.1.1 Sujet de stage

Le stage consiste d'abord à se familiariser aux outils d'optimisation automatique (état de l'art des compilateurs et environnements de compilation itérative) puis à les exploiter sur `CORSIKA` pour obtenir une première version optimisée automatiquement. Dans un second temps il s'agit d'implémenter et évaluer un petit nombre d'optimisations manuelles (préalablement identifiées avec les encadrants) en exploitant entre autre des techniques de vectorisation (instructions SIMD) et en prenant en compte la qualité numérique des calculs. Une analyse comparative des optimisations manuelles et automatiques sera menée à l'aide d'outils spécialisés. Une synthèse mettra en avant des pistes d'amélioration du processus d'optimisation automatique par la prise en compte de la qualité numérique de l'application. Cette recherche se poursuivra dans le cadre d'une thèse.

B.2 Thèse

Afin de poursuivre les travaux démarrés dans le cadre du PEPS Astro-Informatique `CTAOptSim` et du stage de Master d'Adnane Khattabi, en fin 2018 j'ai fait la demande d'une bourse de thèse auprès de l'IN2P3 sur le sujet 'Optimisation des simulations de cascades atmosphériques', décrit dans les sections suivantes. Le sujet de thèse, étant en informatique appliquée à la physique (projet CTA), en concertation avec l'équipe DALI du LIRMM et le groupe CTA du LUPM, nous avons décidé que la thèse s'effectuerait en co-encadrement entre le LUPM et l'équipe DALI du LIRMM :

- Directeurs de thèse : Philippe Langlois (Pr. à l'Université de Perpignan, DALI/LIRMM) et Georges Vasileiadis (DR, LUPM/CNRS)
- Encadrants : Luisa Arrabito (IR, LUPM/CNRS) et David Parello (MdC à l'Université de Perpignan, DALI/LIRMM)

Ayant obtenu une réponse favorable de la part de l'IN2P3, la thèse a pu démarrer en octobre 2019 avec le recrutement de Matthieu Carrère. Matthieu est actuellement inscrit en deuxième année à l'école doctorale I2S de l'Université de Montpellier dans la spécialité informatique. Dès sa première année, Matthieu a obtenu des très bons résultats en portant à terme le travail d'optimisation de `CORSIKA` (cf. Section 4.5). Matthieu a présenté ses résultats au sein du consortium CTA, lors des réunions de l'*Analysis and Simulation Working Group*. La version finale de `CORSIKA` optimisée a été mise en production lors de la dernière campagne de simulations de CTA, pendant l'été 2020. Par ses contributions au travail d'optimisation de `CORSIKA`, Matthieu est également co-auteur de la publication [37] dans le journal *Computing and Software for Big Science*.

B.2.1 Résumé

Le projet CTA (Cherenkov Telescope Array) se propose de construire et d'exploiter le premier observatoire pour les photons gamma de très haute énergie. Les simulations Monte Carlo permettant de caractériser la réponse de l'instrument consomment dès à présent plus de 200 millions d'heures CPU par an et produisent plusieurs Péta-octets de données chaque année. Aujourd'hui, les différentes générations de processeur proposent des améliorations architecturales qui permettent des gains importants en temps d'exécution (x2 à x16) et en consommation d'énergie. Les programmes doivent prendre en compte les différentes spécificités matérielles afin d'en exploiter pleinement les performances. L'objectif de cette thèse est d'optimiser le code de simulation des gerbes atmosphériques de CTA en profitant au maximum des capacités de ces nouvelles générations de processeurs afin de réduire les coûts associés au calcul. Cette thèse s'insère aussi dans le développement d'un nouveau framework de simulation performant destiné aux différents projets internationaux du domaine. Ces travaux contribuent ainsi aux recherches relatives à l'optimisation automatique de performance dans un environnement matériel hétérogène. La thèse s'effectuera dans le cadre d'une collaboration entre les informaticiens de l'équipe DALI/LIRMM à l'Université de Perpignan et les astrophysiciens du LUPM/CNRS (PEPS Astro-Informatique).

B.2.2 Contexte scientifique

Le projet CTA (Cherenkov Telescope Array) a pour objectif la construction et l'exploitation d'un réseau d'une centaine de télescopes Cherenkov situés sur les sites de La Palma (Canaries, Espagne) et Paranal (ESO, Chili). CTA ouvre une nouvelle ère dans l'astronomie gamma de très haute énergie. Un premier grand télescope a été installé à La Palma en 2018 et l'acquisition des premières données est actuellement en cours. Le consortium de CTA regroupe environ 1200 membres, une centaine d'instituts de recherche de 32 pays. Des simulations Monte Carlo détaillées des gerbes induites par les rayons cosmiques et la réponse des télescopes à la lumière Cherenkov, sont nécessaires tout au long du projet afin de calculer les fonctions de réponse des télescopes. C'est à travers ces fonctions de réponse que les astrophysiciens pourront découvrir les sources astronomiques et en déterminer les caractéristiques spatiales, spectrales et temporelles. Pour la simulation des gerbes atmosphériques CTA utilise le logiciel `CORSIKA`, un logiciel de référence dans la communauté des 'rayons cosmiques'. Le temps de calcul associé aux simulations est et restera très important, i.e. environ 200 millions d'heures CPU normalisées par an. Ces productions Monte Carlo utilisent typiquement 8000 cœurs en simultané distribués sur une vingtaine de sites de la grille européenne EGI. La réduction du temps de calcul nécessaire aux simulations est donc un enjeu majeur pour le projet. Un projet de réécriture complète de `CORSIKA` (projet `CORSIKA 8`) a également démarré en 2018, avec l'implication des différentes expériences utilisatrices : CTA, Auger, IceCube, etc. La version actuelle du logiciel `CORSIKA` est en effet issue de développements entamés en 1990. La capacité d'utiliser efficacement les nouvelles générations de processeur est identifiée comme une nécessité à laquelle le nouveau logiciel `CORSIKA 8` devra répondre.

B.2.3 Objectifs

L'objectif de la thèse consiste d'une part à obtenir les performances « optimales » du logiciel `CORSIKA` existant, qui restera en production encore 4-5 ans, et d'autre part à contribuer au développement du framework de `CORSIKA 8`. Concernant l'optimisation de `CORSIKA`, une étude préliminaire réalisée dans le cadre du PEPS Astro-Informatique [1], nous a permis de montrer que les outils d'optimisation automatique présentent des limites lorsqu'ils sont confrontés à des codes complexes comme `CORSIKA`. L'idée directrice est donc d'élargir l'espace des opportunités d'optimisation. Pour cela, nous souhaitons nous affranchir des contraintes liées aux limites des analyses statiques et aux heuristiques des compilateurs, à celles liées aux informations manquantes sur les jeux de données, ainsi qu'à celles liées aux informations manquantes sur la micro-architecture cible. Le doctorant devra d'abord réaliser des transformations « manuelles » sur le programme existant afin de prendre en compte des informations sur les jeux de données et sur la micro-architecture cible. Dans ce processus de transformation, des techniques de vectorisation seront considérées, ainsi que l'utilisation ou le développement de fonctions mathématiques optimisées. Il s'agira ensuite de mesurer et de quantifier les améliorations de performance liées aux différentes contraintes et de mettre en évidence les voies les plus prometteuses pour lever les verrous de l'optimisation automatique. Dans un deuxième temps, le doctorant contribuera au développement du `CORSIKA 8`. L'objectif est de développer un framework général permettant de bénéficier au maximum des performances

1. `https://gite.lirmm.fr/cta-optimization-group/cta-optimization-project/wikis/home`

des architectures modernes et ce de manière transparente. L'idée principale consiste à explorer une nouvelle approche pour la simulation du 'transport des particules', en exploitant plusieurs niveaux de parallélisme, en regroupant des particules avec des propriétés similaires. Cette approche devrait permettre des gains de performance significatifs, qui seront quantifiés dans des cas d'application réalistes. Enfin, la nouvelle version de `CORSIKA` ainsi qu'une première version de `CORSIKA 8` seront déployées sur la grille de calcul EGI pour leur validation complète sur la base d'une statistique élevée. Ce travail s'effectuera dans un contexte international en étroite collaboration avec les astrophysiciens, membres du consortium CTA, ce qui permettra d'assurer la validité physique des résultats obtenus avec la version de `CORSIKA` optimisée et `CORSIKA 8`.

Bibliographie

[1] Science with the cherenkov telescope array. Feb 2018.

[2] A. Casajus et al. Status of the DIRAC project. *J. Phys. Conf. Ser.*, 396 :032107, 2012.

[3] F. Stagni, A. Tsaregorodtsev, L. Arrabito, A. Sailer, T. Hara, and X. Zhang. DIRAC in Large Particle Physics Experiments. *J. Phys. Conf. Ser.*, 898(9) :092020, 2017.

[4] L. Arrabito et al. Application of the DIRAC framework to CTA : First evaluation. *J. Phys. Conf. Ser.*, 396 :032007, 2012.

[5] David Carreto Fidalgo. *Cherenkov Telescopes and MAGIC*, pages 49–81. Springer International Publishing, Cham, 2019.

[6] Heinrich J. Völk and Konrad Bernlöhr. Imaging Very High Energy Gamma-Ray Telescopes. *Exper. Astron.*, 25 :173–191, 2009.

[7] `https://www.cta-observatory.org/project/structure/`.

[8] Martin Barisits, Thomas Beermann, Frank Berghaus, Brian Bockelman, Joaquin Bogado, David Cameron, Dimitrios Christidis, Diego Ciangottini, Gancho Dimitrov, Markus Elsing, Vincent Garonne, Alessandro di Girolamo, Luc Goossens, Wen Guan, Jaroslav Guenther, Tomas Javurek, Dietmar Kuhn, Mario Lassnig, Fernando Lopez, Nicolo Magini, Angelos Molfetas, Armin Nairz, Farid Ould-Saada, Stefan Prenner, Cedric Serfon, Graeme Stewart, Eric Vaand ering, Petya Vasileva, Ralph Vigne, and Tobias Wegner. Rucio : Scientific data management. *Computing and Software for Big Science*, 3(1) :11, Aug 2019.

[9] Adrian Casajus, Ricardo Graciani, Stuart Paterson, and Andrei Tsaregorodtsev. DIRAC pilot framework and the DIRAC workload management system. *J. Phys. Conf. Ser.*, 219 :062049, 2010.

[10] F. Stagni, A. McNab, C. Luzzi, and W. Krzemien. DIRAC universal pilots. *J. Phys. Conf. Ser.*, 898(9) :092024, 2017.

[11] J.T. Mościcki, M. Lamanna, M. Bubak, and P.M.A. Sloot. Processing moldable tasks on the grid : Late job binding with lightweight user-level overlay. *Future Generation Computer Systems*, 27(6) :725 – 736, 2011.

[12] Vincent Garonne. *Étude, définition et modèlisation d'un Système Distribué à Grande Échelle : DIRAC - Distributed Infrastructure with Remote Agent Control.* PhD thesis, U. Mediterranee, Aix-Marseille II, 2005.

[13] R. Fielding. Architectural Styles and the Design of Network-based Software Architectures ; Doctoral dissertation. 2000.

[14] J. Closier et al. Results of the LHCb experiment Data Challenge 2004. 10 2004.

[15] https://indico.cern.ch/event/811997/attachments/1862943/3062278/HPCLHCC.docx.pdf.

[16] Sorina Camarasu-Pop, Carole Lartizien, Thomas Grenier, Axel Bonnet, Pascal Wassong, Vanessa Hamar, Fabio Hernandez, Luisa Arrabito, Johan Bregeon, Pierre Gay, and Andrei Tsaregorodtsev. Exploiting GPUs on distributed infrastructures for medical imaging applications with VIP and DIRAC. In *42nd international convention on information and communication technology, electronics and microelectronics*, pages 190–195, Opatija, Croatia, May 2019.

[17] DataCloud Collaboration, Davide Salomoni, Isabel Campos, Luciano Gaido, Jesus Marco de Lucas, Peter Solagna, Jorge Gomes, Ludek Matyska, Patrick Fuhrman, Marcus Hardt, Giacinto Donvito, Lukasz Dutka, Marcin Plociennik, Roberto Barbera, Ignacio Blanquer, Andrea Ceccanti, Mario David, Cristina Duma, Alvaro Lopez-Garc German Molto, Pablo Orviz, Zdenek Sustr, Matthew Viljoen, Fernando Aguilar, Luis Alves, Marica Antonacci, Lucio Angelo Antonelli, Stefano Bagnasco, Alexandre M. J. J. Bonvin, Riccardo Bruno, Eva Cetinic, Yin Chen, Fabrizio Chiarello, Alessandro Costa, Stefano Dal Pra, Davor Davidovic, Alvise Dorigo, Benjamin Ertl, Federica Fanzago, Marco Fargetta, Sandro Fiore, Stefano Gallozzi, Zeynep Kurkcuoglu, Lara Lloret, Joao Martins, Alessandra Nuzzo, Paola Nassisi, Cosimo Palazzo, Joao Pina, Eva Sciacca, Matteo Segatta, Massimo Sgaravatto, Daniele Spiga, Sonia Taneja, Marco Antonio Tangaro, Michal Urbaniak, Sara Vallero, Marco Verlato, Bas Wegh, Valentina Zaccolo, Federico Zambelli, Lisa Zangrando, Stefano Zani, and Tomasz Zok. Indigo-datacloud :a data and computing platform to facilitate seamless access to e-infrastructures, 2019.

[18] https://www.hnscicloud.eu/sites/default/files/D6.3_Roadmap_for_the_implementation_of_a_full-scale_European_Open_Science_Cloud_v1.1_0.pdf.

[19] A. McNab, F. Stagni, and M.Ubeda Garcia. Running Jobs in the Vacuum. *J. Phys. Conf. Ser.*, 513 :032065, 2014.

[20] A. McNab, P. Love, and E. MacMahon. Managing virtual machines with Vac and Vcycle. *J. Phys. Conf. Ser.*, 664(2) :022031, 2015.

[21] Ryan P. Taylor, Cristovao Jose Domingues Cordeiro, Domenico Giordano, John Hover, Tomas Kouba, Peter Love, Andrew McNab, Jaroslava Schovancova, and Randall Sobie. Consolidation of cloud computing in ATLAS. *J. Phys. Conf. Ser.*, 898(5) :052008, 2017.

[22] A. McNab, F. Stagni, and C. Luzzi. LHCb experience with running jobs in virtual machines. *J. Phys. Conf. Ser.*, 664(2) :022030, 2015.

[23] Rafael Ferreira da Silva, Rosa Filgueira, Ilia Pietri, Ming Jiang, Rizos Sakellariou, and Ewa Deelman. A characterization of workflow management systems for extreme-scale applications. *Future Generation Computer Systems*, 75 :228 – 238, 2017.

[24] L. Arrabito, J. Bregeon, A. Haupt, R. Graciani Diaz, F. Stagni, and A. Tsaregorodtsev. Prototype of a production system for Cherenkov Telescope Array with DIRAC. *J. Phys. Conf. Ser.*, 664(3) :032001, 2015.

[25] F. Stagni and P. Charpentier. The LHCb DIRAC-based production and data management operations systems. *J. Phys. Conf. Ser.*, 368 :012010, 2012.

[26] Luisa Arrabito et al. The Cherenkov Telescope Array production system for data-processing and Monte Carlo simulation. *EPJ Web Conf.*, 214 :03052, 2019.

[27] Luisa Arrabito et al. DIRAC framework evaluation for the Fermi-LAT and CTA experiments. *J. Phys. Conf. Ser.*, 513 :032003, 2014.

[28] L. Arrabito, J. Bregeon, A. Haupt, R. Graciani Diaz, F. Stagni, and A. Tsaregorodtsev. Prototype of a production system for Cherenkov Telescope Array with DIRAC. *J. Phys. Conf. Ser.*, 664(3) :032001, 2015.

[29] L. Arrabito et al. The Cherenkov Telescope Array production system for Monte Carlo simulations and analysis. *J. Phys. Conf. Ser.*, 898(5) :052013, 2017.

[30] F. Stagni, A. Tsaregorodtsev, L. Arrabito, A. Sailer, T. Hara, and X. Zhang. DIRAC in Large Particle Physics Experiments. *J. Phys. Conf. Ser.*, 898(9) :092020, 2017.

[31] Ralph Engel, Dieter Heck, Tim Huege, Tanguy Pierog, Maximilian Reininghaus, Felix Riehn, Ralf Ulrich, Michael Unger, and Darko Veberič. Towards a Next Generation of CORSIKA : A Framework for the Simulation of Particle Cascades in Astroparticle Physics. *Comput. Softw. Big Sci.*, 3(1) :2, 2019.

[32] Gernot Maier, L. Arrabito, K. Bernlöhr, J. Bregeon, F. Di Pierro, T. Hassan, T. Jogler, J. Hinton, A. Moralejo, and M. Wood. Monte Carlo Performance Studies of Candidate Sites for the Cherenkov Telescope Array. *PoS*, ICRC2015 :713, 2016.

[33] Tarek Hassan, Luisa Arrabito, Konrad Bernlör, Johan Bregeon, James Hinton, Tobias Jogler, Gernot Maier, Abelardo Moralejo, Federico Di Pierro, and Matthew Wood. Second large-scale Monte Carlo study for the Cherenkov Telescope Array. *PoS*, ICRC2015 :971, 2016.

[34] T. Hassan et al. Monte Carlo Performance Studies for the Site Selection of the Cherenkov Telescope Array. *Astropart. Phys.*, 93 :76–85, 2017.

[35] A. Acharyya et al. Monte Carlo studies for the optimisation of the Cherenkov Telescope Array layout. *Astropart. Phys.*, 111 :35–53, 2019.

[36] Luisa Arrabito, Konrad Bernlöhr, Johan Bregeon, Gernot Maier, Philippe Langlois, David Parello, and Guillaume Revy. Performance optimization of the air shower simulation program for the Cherenkov Telescope Array. *EPJ Web Conf.*, 214 :05041, 2019.

[37] Luisa Arrabito, Konrad Bernlöhr, Johan Bregeon, Matthieu Carrère, Adnane Khattabi, Philippe Langlois, David Parello, and Guillaume Revy. Optimizing Cherenkov Photons Generation and Propagation in CORSIKA for CTA Monte–Carlo Simulations. *Comput. Softw. Big Sci.*, 4(1) :9, 2020.

[38] M. Kobal. A thinning method using weight limitation for air-shower simulations. *Astropart. Phys.*, 15 :259–273, 2001.

[39] *Architecture software developer's manual volume 1. 2016.*

[40] Christoph Lauter. A new open-source SIMD vector libm fully implemented with high-level scalar C. In *2016 50th Asilomar Conference on Signals, Systems and Computers* , pages 407 – 411, Pacific Grove, United States, November 2016.

[41] G Amadio et al. GeantV : Results from the prototype of concurrent vector particle transport simulation in HEP. 5 2020.

[42] Marko Bracko, Marco Clemencic, Dave Dykstra, Andrea Formica, Giacomo Govi, Michel Jouvin, David Lange, Paul Laycock, and Lynn Wood. HEP Software Foundation Community White Paper Working Group – Conditions Data. 1 2019.

[43] B. P. Abbott, R. Abbott, T. D. Abbott, F. Acernese, K. Ackley, C. Adams, T. Adams, P. Addesso, R. X. Adhikari, V. B. Adya, and et al. Multi-messenger observations of a binary neutron star merger. *The Astrophysical Journal*, 848(2) :L12, Oct 2017.