

Apprehending heterogeneity at (very) large scale

PhD thesis defense

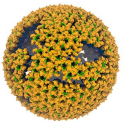
Raphaël Bleuse
under the supervision of
Grégory Mounié & Denis Trystram



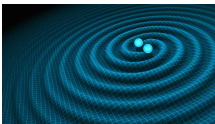
11th October 2017

High Performance Computing: An Ecosystem

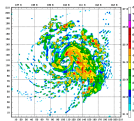
Applications



virology

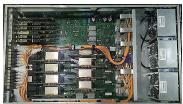


physics



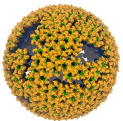
weather forecast

System

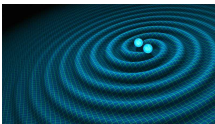


High Performance Computing: An Ecosystem

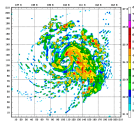
Applications



virology



physics



weather forecast

Middleware

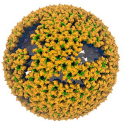
OS, run-time, I/O layers, deployment tools, monitoring tools, RJMS, ...

System

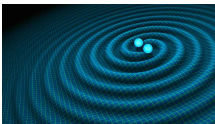


High Performance Computing: An Ecosystem

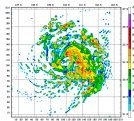
Applications



virology



physics



weather forecast

Middleware

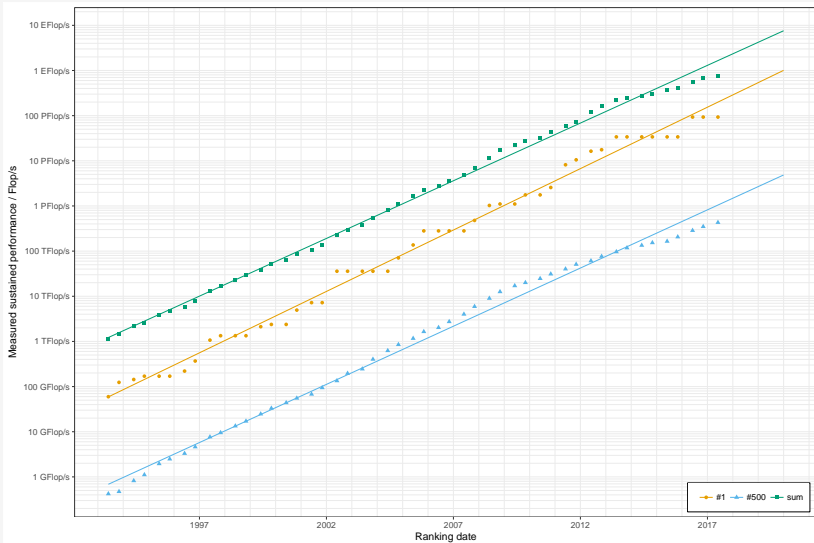
OS, run-time, I/O layers, deployment tools, monitoring tools, **RJMS**, ...

System



Exascale (10^{18}) Goal

Evolution of processing power [@top500]



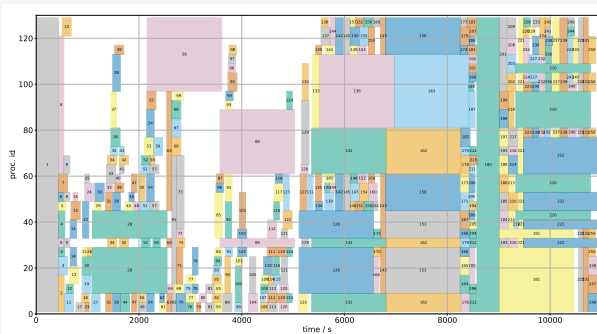
Exascale (10^{18}) Challenges

“Exascale \neq 1000 \times Petascale” [Ash+10; Don+11]

power constraint (20–40 MW limit)

- **accelerators** \rightarrow **heterogeneity**
- **data movements** \rightarrow **new objective/constraint**
 - programming models
 - reliability
 - ...

Scheduling w.r.t. Execution Context



Scheduling Model in HPC

platform

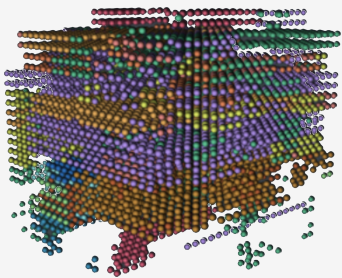
tasks/jobs

objective

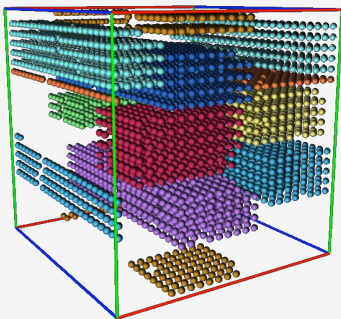
⇒ where? & when?

Topology matters!

Allocation example: snapshot of the 10 biggest jobs on Blue Waters



25.40 s/iter.



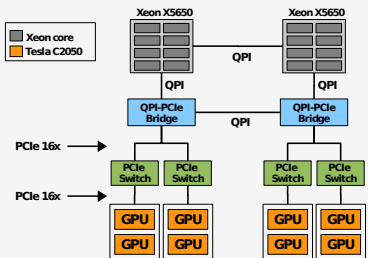
11.64 s/iter.

[Eno+14; PML15; Yil+16]

- fragmentation hinders performances
- convexity helps mitigating interferences

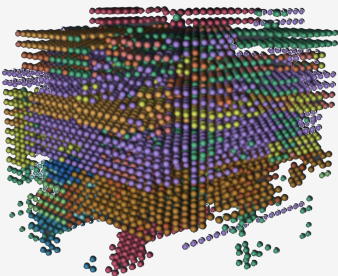
How to bring
at a reasonable cost
context-awareness, or even context-obliviousness,
in the scheduling policies?

Two Axes of Contribution: Intra/Inter-Applications



Intra-Application Axis

- affinity mechanism [Ble+14; Ble+15]
- implicit parallelism on CPUs [Ble+17]



Inter-Applications Axis

- geometric model of HPC platforms
- study of 1D instantiations

1 Introduction

- HPC Ecosystem
- Towards Exascale
- Scheduling for HPC

2 Intra-Application Axis: Affinity

- Problem Formalization
- Performance Evaluation

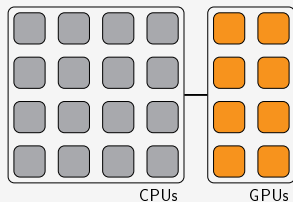
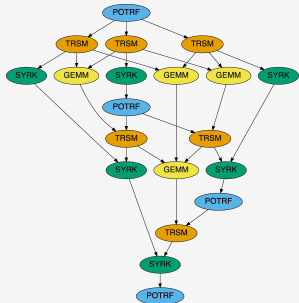
3 Inter-Applications Axis: Modeling HPC Platforms

- Platform Example: Blue Waters
- Modeling Platforms
- 1D Instantiation

4 Conclusion

- Summary
- Perspectives

Problem Formalization



Basic model

- data-flow, batch of ready tasks:
 $\implies n$ independent tasks
- 2 sets of identical nodes:
 m CPUs, k GPUs
- minimize C_{\max}

Integrating Context-Awareness: Affinity

Guiding scheduling decisions w.r.t. Context

New mechanism: Affinity scoring

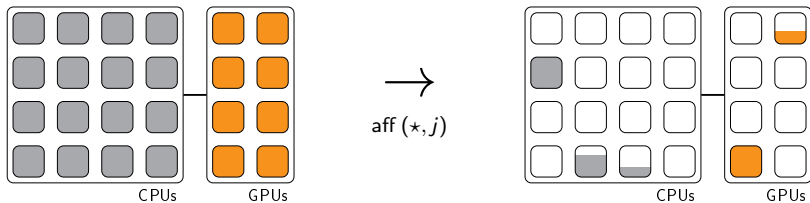
$$\text{aff} : \text{node} \times \text{task} \mapsto \mathbb{R}^+$$

Integrating Context-Awareness: Affinity

Guiding scheduling decisions w.r.t. Context

New mechanism: Affinity scoring

$$\text{aff} : \text{node} \times \text{task} \mapsto \mathbb{R}^+$$

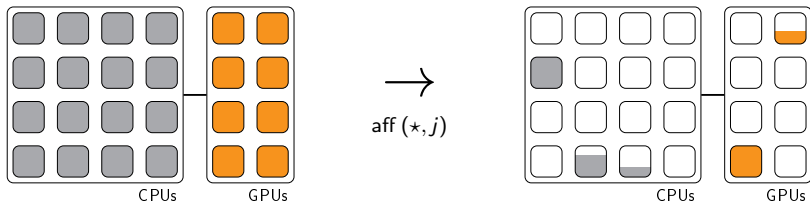


Integrating Context-Awareness: Affinity

Guiding scheduling decisions w.r.t. Context

New mechanism: Affinity scoring

$$\text{aff} : \text{node} \times \text{task} \mapsto \mathbb{R}^+$$



→ maximize score over the allocation (local)
& minimize C_{\max} (global)

Proposed Algorithm: DADA

affinity



Stein & Wein schema [SW97]

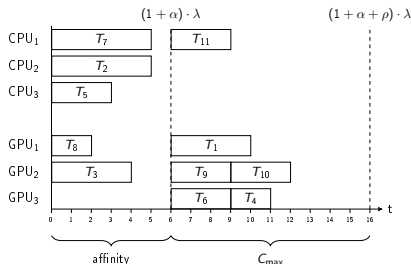


base model

- dual approximation [HS87]
- ρ -approx. ($\rho = 2$)

DADA: Distributed Affinity Dual Approximation

- α : length of affinity phase
- λ : C_{\max}^* guess
→ binary search on λ



Performance Evaluation

Implementation Details / Benchmark Setup

Implementation details

- implemented in XKaapi core (33,385 l.o.c. / scheduler ~ 1000 l.o.c.)
- online computation of affinity: *number of valid bytes*
- 3 variants: DADA($\alpha = 0$), DADA(α), DADA(α)+CP

Performance Evaluation

Implementation Details / Benchmark Setup

Implementation details

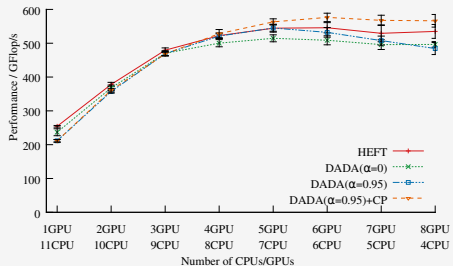
- implemented in XKaapi core (33,385 l.o.c. / scheduler ~ 1000 l.o.c.)
- online computation of affinity: *number of valid bytes*
- 3 variants: DADA($\alpha = 0$), DADA(α), DADA(α)+CP

Benchmark Setup

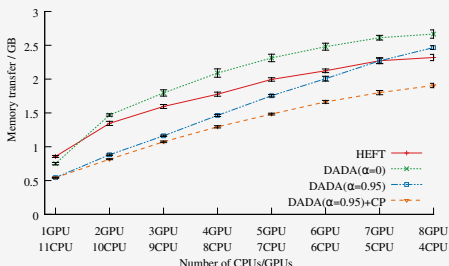
- PLASMA kernels (LU, QR & *Cholesky factorization*)
- scalability in number of GPUs
- 2 metrics:
 - 1 raw performance (GFlop/s) $\rightarrow C_{\max}$
 - 2 data transfers (GB) \rightarrow affinity

Performance Evaluation

Cholesky factorization: 8192×8192 matrices



Performance / GFlop/s
higher is better



Data Transfers / GB
lower is better

1 Introduction

- HPC Ecosystem
- Towards Exascale
- Scheduling for HPC

2 Intra-Application Axis: Affinity

- Problem Formalization
- Performance Evaluation

3 Inter-Applications Axis: Modeling HPC Platforms

- Platform Example: Blue Waters
- Modeling Platforms
- 1D Instantiation

4 Conclusion

- Summary
- Perspectives

Platform Example: Blue Waters

System Summary

- operated by NCSA @ UIUC
- 26,868 computer nodes ; 396,032 cores \implies 13.34 PFlop/s peak
- 672 I/O nodes

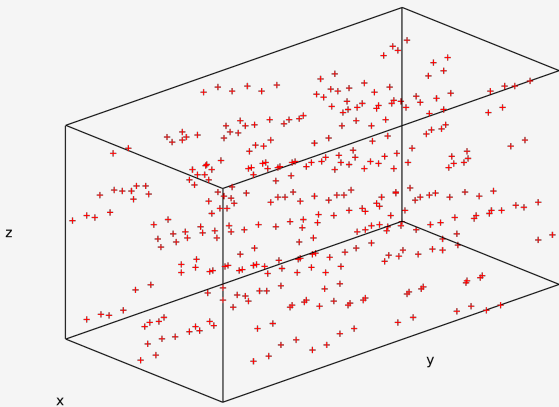
Interconnect

- 3D Torus: $24 \times 24 \times 24$
- single, multi-purpose network
- static dimension-order routing ($x > y > z$)

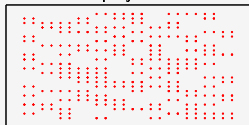


Blue Waters' Architecture

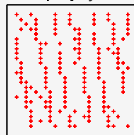
I/O nodes distribution



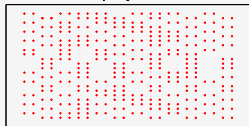
proj. x



proj. y



proj. z



Model

Key Properties recap.

- unique & multi-purpose interconnection network
- heterogeneous nodes (compute & I/O)

Formalization

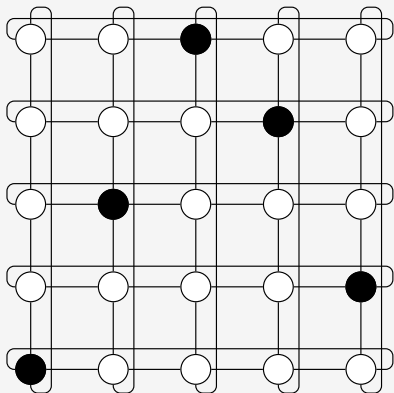
- machine**
- n nodes divided into 2 sets: \mathcal{V}^C & $\mathcal{V}^{I/O}$
 - arbitrary ordering of the nodes: $1 \dots n$
 - interconnection topology
 - distribution: mapping of the nodes on the topology
- jobs**
- set \mathcal{J} of independent jobs
 - p_j : processing time
 - q_j^C : number of requested nodes in \mathcal{V}^C
 - $q_j^{I/O}$: requested nodes in $\mathcal{V}^{I/O}$ (*pinned, unpinned*)

Nocuous Interactions

Types of nodes:

○ compute nodes

● I/O nodes



2D torus, static routing

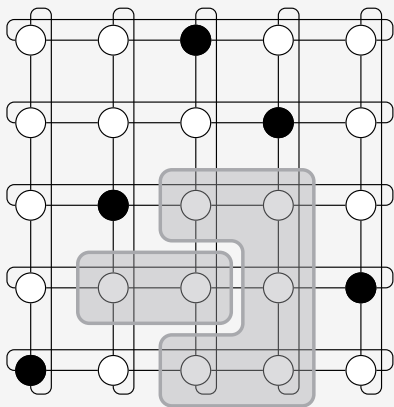
Nocuous Interactions

Types of nodes:

○ compute nodes

● I/O nodes

□ allocation



2D torus, static routing

Nocuous Interactions

Types of nodes:

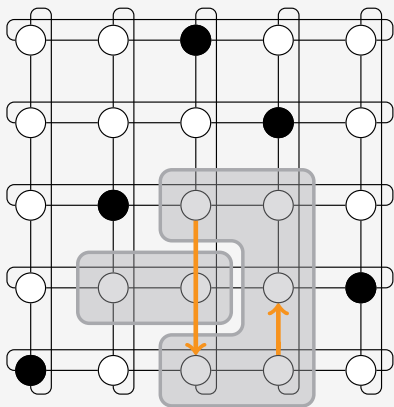
○ compute nodes

● I/O nodes

□ allocation

Communication types:

→ compute comm.



2D torus, static routing

Nocuous Interactions

Types of nodes:

○ compute nodes

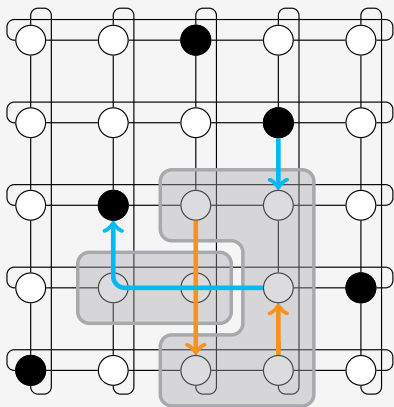
● I/O nodes

□ allocation

Communication types:

→ compute comm.

→ I/O comm.

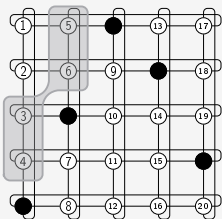


2D torus, static routing

Avoiding Nocuous Interactions

Proposed Intrinsic Geometric Constraints

contiguity allocated nodes form a contiguous range w.r.t. the ordering

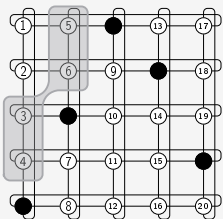


contiguous

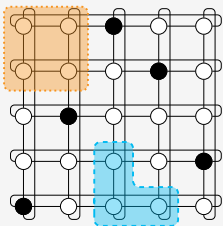
Avoiding Nocuous Interactions

Proposed Intrinsic Geometric Constraints

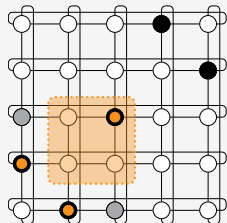
- contiguity** allocated nodes form a contiguous range w.r.t. the ordering
- connectivity** connected component of the topology
- convexity** compute communications cannot be shared
- locality** I/O nodes are adjacent to compute nodes



contiguous



connected, convex



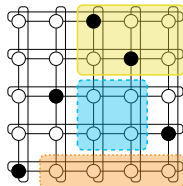
local

Characterizing Allocations

Proposed Extrinsic Metrics

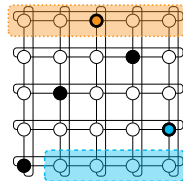
γ -compactness: how spread is an allocation? [PML14]

$$\frac{1}{q_j(q_j - 1)} \sum_{i \in \mathcal{V}^C(j)} \sum_{i' \in \mathcal{V}^C(j)} \text{dist}(i, i')$$



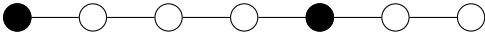
ρ -proximity: how far are I/O nodes?

$$\max_{i \in \mathcal{V}^{I/O}(j)} \frac{1}{q_j} \left(\min_{i' \in \mathcal{V}^C(j)} \text{dist}(i, i') + \max_{i' \in \mathcal{V}^C(j)} \text{dist}(i, i') \right)$$



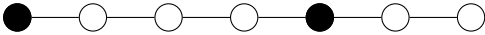
1D Instantiation: *pinned I/O*

Model

- 1D line 
- *pinned I/O*, $|q_j^{I/O}| = 1$
- convex & local
- minimize C_{\max}

1D Instantiation: *pinned I/O*

Model

- 1D line 
- *pinned I/O*, $|q_j^{I/O}| = 1$
- convex & local
- minimize C_{\max}

6-Approximation Algorithm

2 Phases:

- 1 fix the nodes allocated to each jobs (2-approximation)
- 2 schedule the jobs (3-approximation)

[Phase 1] Dedicating nodes: Linear Program

Assign $x_{j,s}$ such that:

$$\begin{aligned} \min \Lambda, \\ \text{s.t. } \Lambda \geq L_i \quad \forall i \quad (C_1) \end{aligned}$$

$$L_i \geq \sum_j \sum_s x_{j,s} p_j \mathbf{1}_{i \in \alpha^C(s, q_j^C)} \quad \forall i \quad (C_2)$$

$$\sum_s x_{j,s} = 1 \quad \forall j \quad (C_3)$$

$$\text{dist}(\alpha^C(s, q_j^C), \mathcal{V}^{I/O}(j)) \leq 1 \quad \forall j \forall s \quad (C_4)$$

$$x_{j,s} \in \{0, 1\} \quad \forall j \forall s \quad (C_5)$$

Λ : global load | L_i : load of node i

p_j : processing time of job j | $\mathcal{V}^{I/O}(j)$: I/O node requested by job j

$\alpha^C(s, q)$: allocation $[s, \dots]$ with q compute nodes

[Phase 1] Dedicating nodes: Linear Program

Assign $x_{j,s}$ such that:

$$\begin{aligned} \min \Lambda, \\ \text{s.t. } \Lambda \geq L_i \quad \forall i \quad (C_1) \end{aligned}$$

$$L_i \geq \sum_j \sum_s x_{j,s} p_j \mathbf{1}_{i \in \alpha^C(s, q_j^C)} \quad \forall i \quad (C_2)$$

$$\sum_s x_{j,s} = 1 \quad \forall j \quad (C_3)$$

$$\text{dist}(\alpha^C(s, q_j^C), \mathcal{V}^{I/O}(j)) \leq 1 \quad \forall j \forall s \quad (C_4)$$

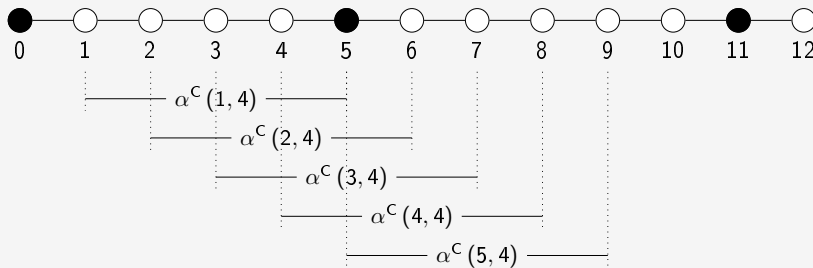
$$0 \leq x_{j,s} \leq 1 \quad \forall j \forall s \quad (C_5)$$

Λ : global load | L_i : load of node i

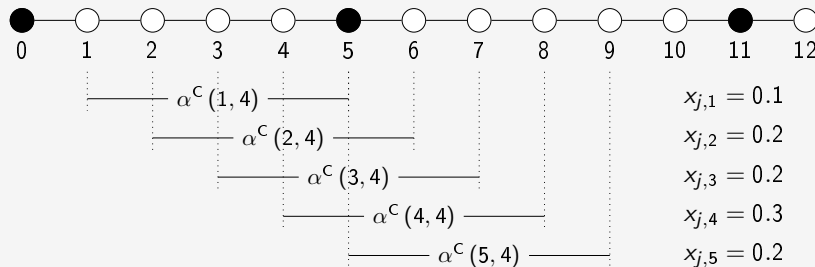
p_j : processing time of job j | $\mathcal{V}^{I/O}(j)$: I/O node requested by job j

$\alpha^C(s, q)$: allocation $[s, \dots]$ with q compute nodes

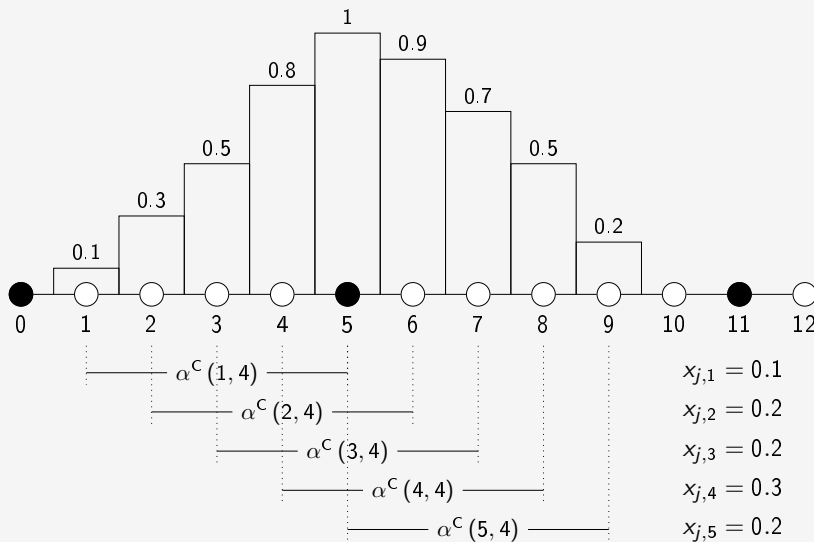
[Phase 1] Dedicating nodes: Rounding



[Phase 1] Dedicating nodes: Rounding



[Phase 1] Dedicating nodes: Rounding



[Phase 1] Dedicating nodes: 2-Approximation

Lemma

The dedication phase finds an allocation for each job such that the maximum load is at most twice the maximum load of the optimal allocation.

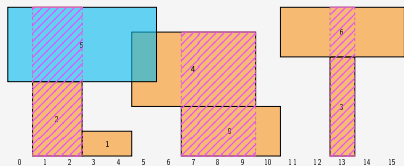
Proof.

Given a job j of unitary load:

- $\mathcal{V}^{I/O}(j)$ adjacent to all valid allocations $\implies L_{\mathcal{V}^{I/O}(j)} = 1$
- $\exists i, L_i > 0.5$
- $\exists s / \forall i \in \alpha^C(s, q_j^C), L_i > 0.5$



[Phase 2] Scheduling: Gergov's Algorithm [Ger99]



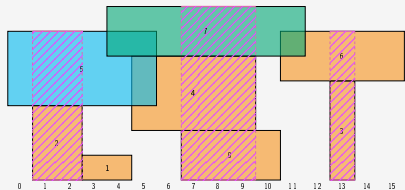
Invariants:

- 1 a job belongs to a *single* pillar
- 2 overlap restricted to direct neighbors
- 3 pillars are *lower bounds* of load

Operations:

- 1 dock job on pillar
- 2 remove pillar

[Phase 2] Scheduling: Gergov's Algorithm [Ger99]



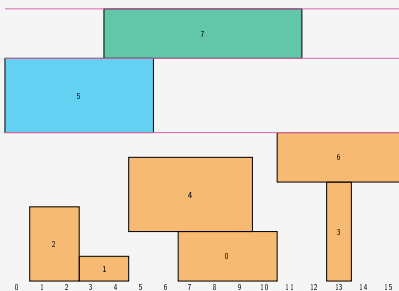
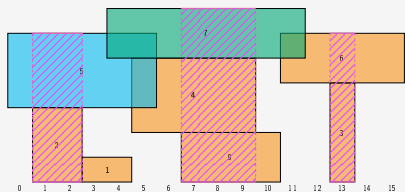
Invariants:

- 1 a job belongs to a *single* pillar
- 2 overlap restricted to direct neighbors
- 3 pillars are *lower bounds* of load

Operations:

- 1 dock job on pillar
- 2 remove pillar

[Phase 2] Scheduling: Gergov's Algorithm [Ger99]



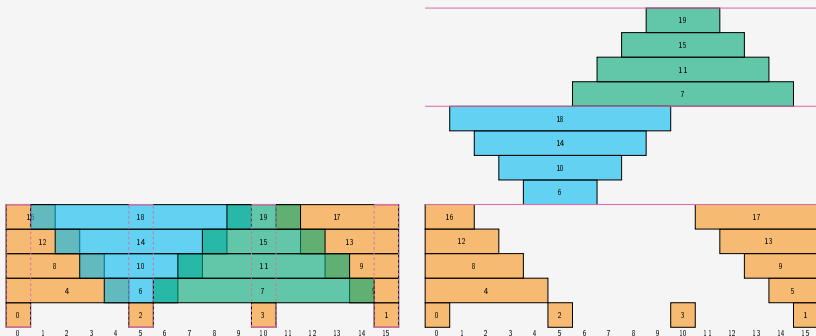
Invariants:

- 1 a job belongs to a *single* pillar
- 2 overlap restricted to direct neighbors
- 3 pillars are *lower bounds* of load

Operations:

- 1 dock job on pillar
- 2 remove pillar

[Phase 2] Scheduling: Gergov's Algorithm Tightness



Theorem (Thesis, p. 91)

The 3-approximation ratio is asymptotically tight.

1 Introduction

- HPC Ecosystem
- Towards Exascale
- Scheduling for HPC

2 Intra-Application Axis: Affinity

- Problem Formalization
- Performance Evaluation

3 Inter-Applications Axis: Modeling HPC Platforms

- Platform Example: Blue Waters
- Modeling Platforms
- 1D Instantiation

4 Conclusion

- Summary
- Perspectives

Summary

Intra-Application Axis

- designed new algorithm with competitive ratio
 - implemented in a run-time
 - conducted performance evaluation
- ⇒ integrated new constraints with maintained performances

Inter-Applications Axis

- new modeling framework
- tackle interference (via reasonable constraints)
- preliminary analysis shows viability

Perspectives

Intra-Application Axis

- vary the affinity function
- integrating other constraints, objectives (e.g., energy, NUMA effects)

Inter-Applications Axis

- single phase algorithm for 1D
- higher dimensions through graph embedding
- integration in production-grade schedulers
- heterogeneity of applications

References I

- Erich Strohmaier et al. *TOP500 list*. url: <https://www.top500.org/lists/> (visited on 06/16/2017).
- Steve Ashby et al. *Opportunities and Challenges of Exascale Computing*. Tech. rep. U.S. Department of Energy, 2010.
- Raphaël Bleuse et al. “Scheduling Data Flow Program in XKaapi: A New Affinity Based Algorithm for Heterogeneous Architectures”. In: *Euro-Par*. Vol. 8632. Lecture Notes in Computer Science. Springer, Aug. 2014, pp. 560–571.
- Raphaël Bleuse et al. “Scheduling independent tasks on multi-cores with GPU accelerators”. In: *Concurrency and Computation: Practice and Experience* 27.6 (2015), pp. 1625–1638.
- Raphaël Bleuse et al. “Scheduling Independent Moldable Tasks on Multi-Cores with GPUs”. In: *IEEE Transactions on Parallel and Distributed Systems* 28.9 (Sept. 2017), pp. 2689–2702.

References II

- Jack Dongarra et al. “The International Exascale Software Project roadmap”. In: *International Journal of High Performance Computing Applications* 25.1 (Jan. 2011), pp. 3–60.
- Jeremy Enos et al. “Topology-Aware Job Scheduling Strategies for Torus Networks”. In: *Cray User Group*. May 2014.
- Jordan Gergov. “Algorithms for Compile-Time Memory Optimization”. In: *SODA*. ACM/SIAM, Jan. 1999, pp. 907–908.
- Dorit S. Hochbaum and David B. Shmoys. “Using Dual Approximation Algorithms for Scheduling Problems: Theoretical and Practical Results”. In: *Journal of the ACM* 34.1 (Jan. 1987), pp. 144–162.
- Jose Antonio Pascual, José Miguel-Alonso, and José Antonio Lozano. “Application-aware metrics for partition selection in cube-shaped topologies”. In: *Parallel Computing* 40.5 (May 2014), pp. 129–139.

References III

- Jose Antonio Pascual, José Miguel-Alonso, and José Antonio Lozano. “Locality-aware policies to improve job scheduling on 3D tori”. In: *The Journal of Supercomputing* 71.3 (Mar. 2015), pp. 966–994.
- Clifford Stein and Joel Wein. “On the existence of schedules that are near-optimal for both makespan and total weighted completion time”. In: *Operations Research Letters* 21.3 (Oct. 1997), pp. 115–122.
- Orcun Yildiz et al. “On the Root Causes of Cross-Application I/O Interference in HPC Storage Systems”. In: *IPDPS*. IEEE, May 2016, pp. 750–759.