Centralized versus distributed schedulers for multiple bag-of-task applications


Laboratoire LaBRI, CNRS Bordeaux, France

Dept. of Computer Science and Engineering, University of California, San Diego, USA

Laboratoire ID-IMAG, CNRS-INRIA Grenoble, France

Laboratoire de l’Informatique du Parallélisme 
École Normale Supérieure de Lyon, France

Graal Working Group – October 2005
Multiple applications:

- competing for CPU and network resources
- consisting in large number of identical independent tasks

Same size for all tasks of one application

Different communication and computation demands for different applications

Important parameter: \[ \frac{\text{communication size}}{\text{computation size}} \] for one application
Multiple applications:

- competing for CPU and network resources
- consisting in large number of identical independent tasks

- Same size for all tasks of one application
- Different communication and computation demands for different applications

Important parameter: \( \frac{\text{communication size}}{\text{computation size}} \) for one application
Multiple applications:
- competing for CPU and network resources
- consisting in large number of identical independent tasks

Same size for all tasks of one application

Different communication and computation demands for different applications

Important parameter: \( \frac{\text{communication size}}{\text{computation size}} \) for one application
Multiple applications:
- competing for CPU and network resources
- consisting in large number of identical independent tasks

- Same size for all tasks of one application
- Different communication and computation demands for different applications
- Important parameter: $\frac{\text{communication size}}{\text{computation size}}$ for one application
- Target platform: master-worker star network

- Master holds all tasks initially
Introduction – Goals

- Maximize throughput
- Maintain balanced execution between application (fairness)
- Scheduling problems:
  - at master: which applications to which subtree
  - at nodes (tree): which tasks to forward to children
- Objective definition:
  - priority weight: \( w^{(k)} \) for application \( A_k \)
  - throughput: \( \alpha^{(k)} = \frac{\text{number of tasks completed at time } t}{\text{for } A_k} \)
  - MAX-MIN fairness: \( \text{MAXIMIZE } \min_k \left\{ \frac{\alpha^{(k)}}{w^{(k)}} \right\} \).
Maximize throughput

Maintain balanced execution between application (fairness)

Scheduling problems:
- at master: which applications to which subtree
- at nodes (tree): which tasks to forward to children

Objective definition:
- priority weight: $w^{(k)}$ for application $A_k$
- throughput: $\alpha^{(k)} = \frac{\text{number of tasks completed at time } t}{t}$ for $A_k$
- MAX-MIN fairness: MAXIMIZE $\min_k \left\{ \frac{\alpha^{(k)}}{w^{(k)}} \right\}$. 
Introduction – Goals

- Maximize throughput
- Maintain balanced execution between application (*fairness*)
- Scheduling problems:
  - at master: which applications to which subtree
  - at nodes (tree): which tasks to forward to children
- Objective definition:
  - priority weight: $w^{(k)}$ for application $A_k$
  - throughput: $\alpha^{(k)} = \frac{\text{number of tasks completed at time } t}{t}$ for $A_k$
  - MAX-MIN fairness: \( \text{MAXIMIZE } \min_k \left\{ \frac{\alpha^{(k)}}{w^{(k)}} \right\} \).
Maximize throughput

Maintain balanced execution between application (fairness)

Scheduling problems:
  ▶ at master: which applications to which subtree
  ▶ at nodes (tree): which tasks to forward to children

Objective definition:
  ▶ priority weight: \( w^{(k)} \) for application \( A_k \)
  ▶ throughput: \( \alpha^{(k)} = \frac{\text{number of tasks completed at time } t}{t} \) for \( A_k \)
  ▶ MAX-MIN fairness: \( \text{MAXIMIZE } \min_k \left\{ \frac{\alpha^{(k)}}{w^{(k)}} \right\} \).
Centralized strategies
- central scheduler at master
- complete and reliable knowledge of the platform
- compute optimal schedule (Linear Programming formulation)
- convenient for small platform

Decentralized strategies
- more realistic for large scale platforms
- only local information available at each node (neighbors)
- limited memory
- decentralized heuristics
Centralized strategies
- central scheduler at master
- complete and reliable knowledge of the platform
- compute optimal schedule (Linear Programming formulation)
- convenient for small platform

Decentralized strategies
- more realistic for large scale platforms
- only local information available at each node (neighbors)
- limited memory
- decentralized heuristics
1. Platform and Application Model
2. Computing the Optimal Solution
3. Decentralized Heuristics
4. Simulation Results
5. Conclusion & Perspectives
Outline

1. Platform and Application Model
2. Computing the Optimal Solution
3. Decentralized Heuristics
4. Simulation Results
5. Conclusion & Perspectives
Platform Model

- star or tree network
- worker $P_1, \ldots, P_p$ master $P_{\text{master}}$
- parent of $P_u$: $P_{p(u)}$
- bandwidth of link $P_u \rightarrow P_{p(u)}$: $b_u$
- computing speed of $P_u$: $c_u$
- full communication/computation overlap
- single-port model
Platform Model

- star or tree network
- worker \( P_1, \ldots, P_p \) master \( P_{\text{master}} \)
- parent of \( P_u \): \( P_{p(u)} \)
- bandwidth of link \( P_u \rightarrow P_{p(u)} \): \( b_u \)
- computing speed of \( P_u \): \( c_u \)
- full communication/computation overlap
- single-port model
star or tree network
worker $P_1, \ldots, P_p$ master $P_{\text{master}}$

parent of $P_u$, $P_{p(u)}$

bandwidth of link $P_u \rightarrow P_{p(u)}$: $b_u$

computing speed of $P_u$: $c_u$

full communication/computation overlap

single-port model
Platform Model

- star or tree network
- worker $P_1, \ldots, P_p$ master $P_{\text{master}}$
- parent of $P_u$: $P_{p(u)}$
- bandwidth of link $P_u \rightarrow P_{p(u)}$: $b_u$
- computing speed of $P_u$: $c_u$
- full communication/computation overlap
- single-port model
Platform Model

- star or tree network
- worker $P_1, \ldots, P_p$ master $P_{\text{master}}$
- parent of $P_u$: $P_{p(u)}$
- bandwidth of link $P_u \to P_{p(u)}$: $b_u$
- computing speed of $P_u$: $c_u$
- full communication/computation overlap
- single-port model
Platform Model

- star or tree network
- worker $P_1, \ldots, P_p$ master $P_{\text{master}}$
- parent of $P_u$: $P_{p(u)}$
- bandwidth of link $P_u \rightarrow P_{p(u)}$: $b_u$
- computing speed of $P_u$: $c_u$
- full communication/computation overlap
- single-port model
Platform Model

- star or tree network
- worker \( P_1, \ldots, P_p \) master \( P_{\text{master}} \)
- parent of \( P_u: P_{p(u)} \)
- bandwidth of link \( P_u \rightarrow P_{p(u)}: b_u \)
- computing speed of \( P_u: c_u \)
- full communication/computation overlap
- single-port model
$K$ applications $A_1, \ldots, A_k$

- priority weights $w^{(k)}$: $w^{(1)} = 3$ and $w^{(2)} = 1 \iff$ we should process 3 times more $A_1$ than $A_2$
  
- $A_k$ consists in many independent tasks:
  
  ▶ with processing cost $c^{(k)}$ (MFlops)
  ▶ with communication cost $b^{(k)}$ (MBytes)

- communication for data only (no result message)

- communication-to-computation ratio (CCR): $\frac{b^{(k)}}{c^{(k)}}$
Application Model

- \( K \) applications \( A_1, \ldots, A_k \)
- priority weights \( w^{(k)}: w^{(1)} = 3 \) and \( w^{(2)} = 1 \) \( \iff \) we should process 3 times more \( A_1 \) than \( A_2 \)
- \( A_k \) consists in many independent tasks:
  - with processing cost \( c^{(k)} \) (MFlops)
  - with communication cost \( b^{(k)} \) (MBytes)
- communication for data only (no result message)
- communication-to-computation ratio (CCR): \( \frac{b^{(k)}}{c^{(k)}} \)
Applications $A_1, \ldots, A_k$

Priority weights $w^{(k)}$: $w^{(1)} = 3$ and $w^{(2)} = 1 \iff$ we should process 3 times more $A_1$ than $A_2$

$A_k$ consists in many independent tasks:
- with processing cost $c^{(k)}$ (MFlops)
- with communication cost $b^{(k)}$ (MBytes)

Communication for data only (no result message)

Communication-to-computation ratio (CCR): $\frac{b^{(k)}}{c^{(k)}}$
Application Model

- $K$ applications $A_1, \ldots, A_k$
- priority weights $w^{(k)}$: $w^{(1)} = 3$ and $w^{(2)} = 1 \iff$ we should process 3 times more $A_1$ than $A_2$
- $A_k$ consists in many independent tasks:
  - with processing cost $c^{(k)}$ (MFlops)
  - with communication cost $b^{(k)}$ (MBytes)
- communication for data only (no result message)
- communication-to-computation ratio (CCR): $\frac{b^{(k)}}{c^{(k)}}$
Application Model

- $K$ applications $A_1, \ldots, A_k$
- priority weights $w^{(k)}$: $w^{(1)} = 3$ and $w^{(2)} = 1 \iff$ we should process 3 times more $A_1$ than $A_2$
- $A_k$ consists in many independent tasks:
  - with processing cost $c^{(k)}$ (MFlops)
  - with communication cost $b^{(k)}$ (MBytes)
- communication for data only (no result message)
- communication-to-computation ratio (CCR): $\frac{b^{(k)}}{c^{(k)}}$
Computing the Optimal Solution

Linear Program for star network

- $\alpha_u^{(k)}$ = rational number of tasks of $A_k$ executed by $P_u$ every time-unit
- $\alpha_u^{(k)} = 0$ for all $A_k \iff P_u$ does not participate
- constraint for computation at $P_u$:
  \[ \sum_k \alpha_u^{(k)} \cdot c^{(k)} \leq c_u \]
- number of bytes sent to worker $P_u$: \[ \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)} \]
- constraint for communications:
  \[ \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)} \leq 1 \]
- throughput for application $A_k$: \[ \alpha^{(k)} = \sum_{u=1}^{p} \alpha_u^{(k)} \]
- objective:
  \[ \text{MAXIMIZE } \min_k \frac{\alpha^{(k)}}{w^{(k)}} \]
Computing the Optimal Solution

Linear Program for star network

- $\alpha_u^{(k)} = \text{rational number of tasks of } A_k \text{ executed by } P_u \text{ every time-unit}$
- $\alpha_u^{(k)} = 0 \text{ for all } A_k \iff P_u \text{ does not participate}
- \text{constraint for computation at } P_u:\n  \sum_k \alpha_u^{(k)} \cdot c^{(k)} \leq c_u$
- number of bytes sent to worker $P_u$: $\sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)}$
- \text{constraint for communications:}\n  \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)} \leq 1$
- \text{throughput for application } A_k: \alpha^{(k)} = \sum_{u=1}^{p} \alpha_u^{(k)}$
- \text{objective:}\n  \text{MAXIMIZE } \min_k \frac{\alpha^{(k)}}{w^{(k)}}
Linear Program for star network

- $\alpha^{(k)}_u = \text{rational number of tasks of } A_k \text{ executed by } P_u \text{ every time-unit}$
- $\alpha^{(k)}_u = 0$ for all $A_k \iff P_u \text{ does not participate}$
- Constraint for computation at $P_u$:
  \[ \sum_k \alpha^{(k)}_u \cdot c^{(k)} \leq c_u \]
- Number of bytes sent to worker $P_u$: \[ \sum_{k=1}^K \alpha^{(k)}_u \cdot b^{(k)} \]
- Constraint for communications:
  \[ \sum_{k=1}^K \alpha^{(k)}_u \cdot b^{(k)} \leq 1 \]
- Throughput for application $A_k$: $\alpha^{(k)} = \sum_{u=1}^p \alpha^{(k)}_u$
- Objective:
  \[ \text{MAXIMIZE} \min_k \frac{\alpha^{(k)}}{w^{(k)}} \]
Computing the Optimal Solution

Linear Program for star network

- \( \alpha_u^{(k)} \) = rational number of tasks of \( A_k \) executed by \( P_u \) every time-unit
- \( \alpha_u^{(k)} = 0 \) for all \( A_k \) \( \iff \) \( P_u \) does not participate
- constraint for computation at \( P_u \):
  \[
  \sum_k \alpha_u^{(k)} \cdot c(k) \leq c_u
  \]
- number of bytes sent to worker \( P_u \): \( \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b(k) \)
- constraint for communications:
  \[
  \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b(k) \leq 1
  \]
- throughput for application \( A_k \): \( \alpha^{(k)} = \sum_{u=1}^{p} \alpha_u^{(k)} \)
- objective:
  \[
  \text{MAXIMIZE} \ \ \ \ \ \min_k \frac{\alpha^{(k)}}{w^{(k)}}
  \]
Computing the Optimal Solution

Linear Program for star network

- $\alpha_u^{(k)}$ = rational number of tasks of $A_k$ executed by $P_u$ every time-unit
- $\alpha_u^{(k)} = 0$ for all $A_k \iff P_u$ does not participate
- constraint for computation at $P_u$:
  \[ \sum_k \alpha_u^{(k)} \cdot c^{(k)} \leq c_u \]
- number of bytes sent to worker $P_u$: $\sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)}$
- constraint for communications:
  \[ \sum_{u=1}^{p} k=1 \sum_{u=1}^{p} \alpha_u^{(k)} \cdot b^{(k)} \leq 1 \]
- throughput for application $A_k$: $\alpha^{(k)} = \sum_{u=1}^{p} \alpha_u^{(k)}$
- objective:
  \[ \text{MAXIMIZE} \min_k \frac{\alpha^{(k)}}{\omega^{(k)}} \]
Computing the Optimal Solution

**Linear Program for star network**

- $\alpha_u^{(k)}$ = rational number of tasks of $A_k$ executed by $P_u$ every time-unit
- $\alpha_u^{(k)} = 0$ for all $A_k$ $\iff$ $P_u$ does not participate
- constraint for computation at $P_u$:
  \[ \sum_k \alpha_u^{(k)} \cdot c^{(k)} \leq c_u \]
- number of bytes sent to worker $P_u$: $\sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)}$
- constraint for communications:
  \[ \sum_{u=1}^{p} \frac{\sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)}}{b_u} \leq 1 \]
- throughput for application $A_k$: $\alpha^{(k)} = \sum_{u=1}^{p} \alpha_u^{(k)}$
- objective:

\[ \text{MAXIMIZE} \min_k \frac{\alpha^{(k)}}{\omega^{(k)}} \]
Computing the Optimal Solution

Linear Program for star network

- $\alpha_u^{(k)}$ = rational number of tasks of $A_k$ executed by $P_u$ every time-unit
- $\alpha_u^{(k)} = 0$ for all $A_k \iff P_u$ does not participate
- constraint for computation at $P_u$:
  \[ \sum_k \alpha_u^{(k)} \cdot c(k) \leq c_u \]
  number of bytes sent to worker $P_u$: \[ \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b(k) \]
- constraint for communications:
  \[ \sum_{u=1}^{p} \frac{\sum_{k=1}^{K} \alpha_u^{(k)} \cdot b(k)}{b_u} \leq 1 \]
- throughput for application $A_k$: $\alpha^{(k)} = \sum_{u=1}^{p} \alpha_u^{(k)}$
- objective:
  \[
  \text{MAXIMIZE} \quad \min_k \frac{\alpha^{(k)}}{w^{(k)}}
  \]
Reconstructing an Optimal Schedule

- solution of the linear program: \( \alpha_u^{(k)} = \frac{p_{u,k}}{q_{u,k}} \), throughput \( \rho \)
- set the length of the period: \( T_p = \text{lcm}\{q_{u,k}\} \)
- in each period, send \( n_u^{(k)} = \alpha_u^{(k)} \cdot T_{\text{period}} \) to each worker \( P_u \)
- \( \Rightarrow \) periodic schedule with throughput \( \rho \)
- initialization and clean-up phases
- asymptotically optimal schedule (computes the optimal number of tasks in time \( T \), up to a constant \( B \))
Computing the Optimal Solution

Reconstructing an Optimal Schedule

- solution of the linear program: \( \alpha_u^{(k)} = \frac{p_{u,k}}{q_{u,k}} \), throughput \( \rho \)
- set the length of the period: \( T_p = \text{lcm}\{q_{u,k}\} \)
- in each period, send \( n_u^{(k)} = \alpha_u^{(k)} \cdot T_{\text{period}} \) to each worker \( P_u \)
- \( \Rightarrow \) periodic schedule with throughput \( \rho \)
- initialization and clean-up phases
- asymptotically optimal schedule (computes the optimal number of tasks in time \( T \), up to a constant \( B \))
solution of the linear program: $\alpha_u^{(k)} = \frac{p_{u,k}}{q_{u,k}}$, throughput $\rho$

- set the length of the period: $T_p = \text{lcm}\{q_{u,k}\}$

- in each period, send $n_u^{(k)} = \alpha_u^{(k)} \cdot T_{\text{period}}$ to each worker $P_u$

$\Rightarrow$ periodic schedule with throughput $\rho$

- initialization and clean-up phases

- asymptotically optimal schedule (computes the optimal number of tasks in time $T$, up to a constant $B$)
solution of the linear program: $\alpha_u^{(k)} = \frac{p_{u,k}}{q_{u,k}}$, throughput $\rho$

set the length of the period: $T_p = \text{lcm}\{q_{u,k}\}$

in each period, send $n_u^{(k)} = \alpha_u^{(k)} \cdot T_{\text{period}}$ to each worker $P_u$

$\Rightarrow$ periodic schedule with throughput $\rho$

initialization and clean-up phases

asymptotically optimal schedule (computes the optimal number of tasks in time $T$, up to a constant $B$)
Computing the Optimal Solution

Reconstructing an Optimal Schedule

- solution of the linear program: \( \alpha^{(k)}_u = \frac{p_{u,k}}{q_{u,k}} \), throughput \( \rho \)
- set the length of the period: \( T_p = \text{lcm}\{q_{u,k}\} \)
- in each period, send \( n^{(k)}_u = \alpha^{(k)}_u \cdot T_{\text{period}} \) to each worker \( P_u \)
- \( \Rightarrow \) periodic schedule with throughput \( \rho \)
- initialization and clean-up phases
  - asymptotically optimal schedule (computes the optimal number of tasks in time \( T \), up to a constant \( B \))
Computing the Optimal Solution

Reconstructing an Optimal Schedule

- solution of the linear program: \( \alpha_u(k) = \frac{p_{u,k}}{q_{u,k}} \), throughput \( \rho \)
- set the length of the period: \( T_p = \text{lcm}\{q_{u,k}\} \)
- in each period, send \( n_u(k) = \alpha_u(k) \cdot T_{\text{period}} \) to each worker \( P_u \)
- \( \Rightarrow \) periodic schedule with throughput \( \rho \)
- initialization and clean-up phases
- asymptotically optimal schedule (computes the optimal number of tasks in time \( T \), up to a constant \( B \))
Computing the Optimal Solution

Structure of the Optimal Solution

Theorem

- Sort the link by bandwidth so that \( b_1 \geq b_2 \ldots \geq b_p \).
- Sort the applications by CCR so that \( \frac{b^{(1)}}{c^{(1)}} \geq \frac{b^{(2)}}{c^{(2)}} \ldots \geq \frac{b^{(K)}}{c^{(K)}} \).

Then there exist indices \( a_0 \leq a_1 \ldots \leq a_K \), \( a_0 = 1 \), \( a_{k-1} \leq a_k \) for \( 1 \leq k \leq K \), \( a_K \leq p \), such that only processors \( P_u, u \in [a_{k-1}, a_k] \), execute tasks of type \( k \) in the optimal solution.
Adaptation to Tree Networks

- Linear Program can be adapted
- Similarly reconstruct periodic schedule
- No proof of a particular structure
Adaptation to Tree Networks

- Linear Program can be adapted
- Similarly reconstruct periodic schedule
- No proof of a particular structure
Adaptation to Tree Networks

- Linear Program can be adapted
- Similarly reconstruct periodic schedule
- No proof of a particular structure
Problems in previous solutions

- LP approach:
  - centralized, needs all global information at master
  - schedule has possibly huge period
  - → difficult to adapt to load variation
  - big memory requirement
Problems in previous solutions

- LP approach:
  - centralized, needs all global information at master
  - schedule has possibly huge period
  - difficult to adapt to load variation
  - big memory requirement
Problems in previous solutions

- LP approach:
  - centralized, needs all global information at master
  - schedule has possibly huge period
  - → difficult to adapt to load variation
  - big memory requirement
Computing the Optimal Solution

Problems in previous solutions

- LP approach:
  - centralized, needs all global information at master
  - schedule has possibly huge period
  - difficult to adapt to load variation
  - big memory requirement
Outline

1. Platform and Application Model
2. Computing the Optimal Solution
3. Decentralized Heuristics
4. Simulation Results
5. Conclusion & Perspectives
**General scheme for a decentralized heuristic:**

- finite buffer (makes the problem NP hard)
- *demand-driven* algorithms
- local scheduler:

  **Loop**
  
  If there will be room in your buffer, request work from parent.
  
  Select which child to assign work to.
  
  Select the type of application that will be assigned.
  
  Get incoming requests from your local worker and children, if any.
  
  Move incoming tasks from your parent, if any, into your buffer.
  
  **If** you have a task and a request that match your choice **Then**
  
  Send the task to the chosen thread (when the send port is free)
  
  **Else**
  
  Wait for a request or a task

- use only *local* information
Centralized LP based (LP)
- solve linear program with global information
- give each node the $\alpha_u^{(k)}$ for its children and himself
- use a 1D load balancing mechanism with these ratios
- $\rightarrow$ close to optimal throughput?

First Come First Served (FCFS)
- each scheduler enforces a FCFS policy
- master ensures fairness using 1D load balancing mechanism
Decentralized Heuristics

Heuristics – LP

- **Centralized LP based (LP)**
  - solve linear program with global information
  - give each node the $\alpha_u^{(k)}$ for its children and himself
  - use a 1D load balancing mechanism with these ratios
  - $\rightarrow$ close to optimal throughput?

- **First Come First Served (FCFS)**
  - each scheduler enforces a FCFS policy
  - master ensures fairness using 1D load balancing mechanism
Coarse-Grain Bandwidth-Centric (CGBC)

- bandwidth-centric = optimal solution for 1 type of task (send tasks to best communicating child first)
- assemble different types of tasks in one:

\[ w^{(1)} = 3 \quad w^{(2)} = 2 \quad w^{(3)} = 1 \]

- not expected to reach optimal throughput: slow links are used to transfer task with high CCR
Parallel Bandwidth-Centric (PBC)

- superpose bandwidth-centric for each type of task
- on each worker, $K$ independent schedulers
- fairness enforced by the master, distributing the tasks
- independent schedulers $\rightarrow$ concurrent transfers
  limited capacity on the outgoing port
  $\leadsto$ gives an (unfair) advantage to PBC (allows interruptible communications)
Parallel Bandwidth-Centric (PBC)

- superpose bandwidth-centric for each type of task
- on each worker, $K$ independent schedulers
- fairness enforced by the master, distributing the tasks
- independent schedulers $\rightarrow$ concurrent transfers
- limited capacity on the outgoing port
  $\leadsto$ gives an (unfair) advantage to PBC (allows interruptible communications)
Parallel Bandwidth-Centric (PBC)

- superpose bandwidth-centric for each type of task
- on each worker, $K$ independent schedulers
- fairness enforced by the master, distributing the tasks
- independent schedulers $\rightarrow$ concurrent transfers
- limited capacity on the outgoing port
- $\leadsto$ gives an (unfair) advantage to PBC (allows interruptible communications)
Parallel Bandwidth-Centric (PBC)

- superpose bandwidth-centric for each type of task
- on each worker, $K$ independent schedulers
- fairness enforced by the master, distributing the tasks
- independent schedulers $\rightarrow$ concurrent transfers
  limited capacity on the outgoing port
  $\leadsto$ gives an (unfair) advantage to PBC (allows interruptible communications)
Heuristics – DATA-CENTRIC

- **Data-centric scheduling (DATA-CENTRIC)**
  - decentralized heuristic
  - try to convergence to the solution of the LP
  - intuition based on the structure of optimal solution of stars
  - start by scheduling only tasks with higher CCR, then periodically:
    - substitute tasks of type A (high CCR) for tasks of type B (lower CCR)
    - if unused bandwidth appears, send more tasks with high CCR
    - if only tasks with high CCR are sent, lower this quantity to free bandwidth, to send other types of tasks
  - needs information on neighbors
  - some operations are decided on the master, then propagated along the tree
Data-centric scheduling (DATA-CENTRIC)

- decentralized heuristic
- try to convergence to the solution of the LP
- intuition based on the structure of optimal solution of stars
- start by scheduling only tasks with higher CCR, then periodically:
  - substitute tasks of type A (high CCR) for tasks of type B (lower CCR)
  - if unused bandwidth appears, send more tasks with high CCR
  - if only tasks with high CCR are sent, lower this quantity to free bandwidth, to send other types of tasks
- needs information on neighbors
- some operations are decided on the master, then propagated along the tree
Decentralized Heuristics

Heuristics – DATA-CENTRIC

- Data-centric scheduling (DATA-CENTRIC)
  - decentralized heuristic
  - try to convergence to the solution of the LP
  - intuition based on the structure of optimal solution of stars
  - start by scheduling only tasks with higher CCR, then periodically:
    - substitute tasks of type A (high CCR) for tasks of type B (lower CCR)
    - if unused bandwidth appears, send more tasks with high CCR
    - if only tasks with high CCR are sent, lower this quantity to free bandwidth, to send other types of tasks
  - needs information on neighbors
  - some operations are decided on the master, then propagated along the tree
Data-centric scheduling (DATA-CENTRIC)

- decentralized heuristic
- try to convergence to the solution of the LP
- intuition based on the structure of optimal solution of stars
- start by scheduling only tasks with higher CCR, then periodically:
  - substitute tasks of type A (high CCR) for tasks of type B (lower CCR)
  - if unused bandwidth appears, send more tasks with high CCR
  - if only tasks with high CCR are sent, lower this quantity to free bandwidth, to send other types of tasks
- needs information on neighbors
- some operations are decided on the master, then propagated along the tree
Data-centric scheduling (DATA-CENTRIC)

- decentralized heuristic
- try to convergence to the solution of the LP
- intuition based on the structure of optimal solution of stars
- start by scheduling only tasks with higher CCR, then periodically:
  - substitute tasks of type A (high CCR) for tasks of type B (lower CCR)
  - if unused bandwidth appears, send more tasks with high CCR
  - if only tasks with high CCR are sent, lower this quantity to free bandwidth, to send other types of tasks
- needs information on neighbors
- some operations are decided on the master, then propagated along the tree
Outline

1. Platform and Application Model
2. Computing the Optimal Solution
3. Decentralized Heuristics
4. Simulation Results
5. Conclusion & Perspectives
Methodology

- How to measure fair-throughput?
  - concentrate on the phase where all applications are run
    \[ T = \text{earliest time that all tasks of one application are done} \]
  - ignore initialization and termination phases
  - time-interval \([0.1 \times T ; 0.9 \times T]\)
  - compute throughput for each application on this interval

- Platform generation
  - 150 random platforms generated, preferring wide trees
  - links and processors characteristics based on measured values
  - buffer of size 10 tasks (of any type)

- Application generation
  - CCR chosen between 0.001 (matrix multiplication) and 4.6 (matrix addition)

- Heuristic implementation
  - distributed implementation using SimGrid,
  - links and processors capacities measured within SimGrid
Methodology

- How to measure fair-throughput ?
  - concentrate on the phase where all applications are run
    \[ T = \text{earliest time that all tasks of one applications are done} \]
  - ignore initialization and termination phases
    time-interval \([0.1 \times T ; 0.9 \times T]\)
  - compute throughput for each application on this interval

- Platform generation
  - 150 random platforms generated, preferring wide trees
  - links and processors characteristics based on measured values
  - buffer of size 10 tasks (of any type)

- Application generation
  - CCR chosed between 0.001 (matrix multiplication) and 4.6 (matrix addition)

- Heuristic implementation
  - distributed implementation using SimGrid,
  - links and processors capacities measured within SimGrid
Methodology

- How to measure fair-throughput?
  - concentrate on the phase where all applications are run
    - $T =$ earliest time that all tasks of one application are done
  - ignore initialization and termination phases
    - time-interval $[0.1 \times T ; 0.9 \times T]$
  - compute throughput for each application on this interval

- Platform generation
  - 150 random platforms generated, preferring wide trees
  - links and processors characteristics based on measured values
  - buffer of size 10 tasks (of any type)

- Application generation
  - CCR chose between 0.001 (matrix multiplication) and 4.6 (matrix addition)

- Heuristic implementation
  - distributed implementation using SimGrid,
  - links and processors capacities measured within SimGrid
Methodology

- **How to measure fair-throughput?**
  - concentrate on the phase where all applications are run
  \[ T = \text{earliest time that all tasks of one application are done} \]
  - ignore initialization and termination phases
  - time-interval \([0.1 \times T ; 0.9 \times T]\)
  - compute throughput for each application on this interval

- **Platform generation**
  - 150 random platforms generated, preferring wide trees
  - links and processors characteristics based on measured values
  - buffer of size 10 tasks (of any type)

- **Application generation**
  - CCR chose between 0.001 (matrix multiplication) and 4.6 (matrix addition)

- **Heuristic implementation**
  - distributed implementation using SimGrid,
  - links and processors capacities measured within SimGrid
Simulation Results

Theoretical v/ Experimental Throughput

- LP, CGBC: possible to compute expected (theoretical) throughput

![Graph showing frequency vs deviation from theoretical throughput]

- Average deviation = 9.4%

- Increase buffer size from 10 to 200 → average deviation = 0.3%

- In the following, LP = basis for comparison

- Compute $\log \frac{\text{performance of } H}{\text{performance of } LP}$ for each heuristic H, on each platform

- We plot the distribution
Simulation Results

Theoretical v/ Experimental Throughput

- LP, CGBC: possible to compute expected (theoretical) throughput

average deviation $= 9.4\%$

- increase buffer size from 10 to 200 $\rightarrow$ average deviation $= 0.3\%$

- in the following, LP = basis for comparison

- compute $\log \frac{\text{performance of } H}{\text{performance of } LP}$ for each heuristic $H$, on each platform

- we plot the distribution
Simulation Results

Theoretical v/ Experimental Throughput

- LP, CGBC: possible to compute expected (theoretical) throughput

![Diagram](image)

- deviation from theoretical throughput

- average deviation $= 9.4\%$

- increase buffer size from 10 to 200 $\rightarrow$ average deviation $= 0.3\%$

- in the following, LP = basis for comparison

- compute $\log \frac{\text{performance of } H}{\text{performance of LP}}$

- for each heuristic $H$, on each platform

- we plot the distribution
Simulation Results

Theoretical v/ Experimental Throughput

- LP, CGBC: possible to compute expected (theoretical) throughput

\[
\text{average deviation } = 9.4\%
\]

- increase buffer size from 10 to 200 \(\rightarrow\) average deviation = 0.3%

- in the following, LP = basis for comparison
  - compute \(\log\frac{\text{performance of } H}{\text{performance of } \text{LP}}\) for each heuristic \(H\), on each platform
  - we plot the distribution
Simulation Results

Theoretical v/ Experimental Throughput

- LP, CGBC: possible to compute expected (theoretical) throughput

- Increase buffer size from 10 to 200 → average deviation = 0.3%

- In the following, LP = basis for comparison

- Compute $\log \frac{\text{performance of } H}{\text{performance of } \text{LP}}$ for each heuristic H, on each platform

- We plot the distribution

average deviation = 9.4%
Simulation Results

Theoretical v/ Experimental Throughput

- LP, CGBC: possible to compute expected (theoretical) throughput

![Histogram of deviation from theoretical throughput]

- Average deviation = 9.4%

- Increase buffer size from 10 to 200 → average deviation = 0.3%

- In the following, LP = basis for comparison

- Compute \( \log \frac{\text{performance of } H}{\text{performance of } LP} \) for each heuristic H, on each platform

- We plot the distribution
Simulation Results

Performance of FCFS

- geometrical average: FCFS is 1.56 times worse than LP
- worst case: 8 times worse
- geometrical average: CGBC is 1.15 times worse than LP
- worst case: 2 times worse
in 35% of the cases: one application is totally unfavored, its throughput is close to 0.
Performance of DATA-CENTRIC

- geometrical average: DATA-CENTRIC is 1.16 worse than LP
- few instances with very bad solution
- on most platforms, very good solution
- hard to know why it performs bad in few cases
Conclusion & Perspectives

Outline

1. Platform and Application Model
2. Computing the Optimal Solution
3. Decentralized Heuristics
4. Simulation Results
5. Conclusion & Perspectives
Conclusion

Contributions:

- centralized algorithm able to compute optimal solution with global information
- nice characterization of way to compute optimal solution on single-level trees
- design of distributed heuristics to deal with practical settings of Grids (distributed information, variability, limited memory)
- evaluation of these heuristics through extensive simulations
- good performance of sophisticated heuristics compared to the optimal scheduling
Adapt the decentralized computation of MultiCommodity Flow (Awerbuch & Leighton) to our problem
- decentralized approach to compute optimal throughput
- slow convergence speed

Consider other kinds of fairness: proportional fairness
- reasonable (close to the behavior of TCP)
- easy to realize with distributed algorithms