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

November 2005
Large-scale distributed platforms result from the collaboration of many users:

- sharing resources amongst users should somehow be fair.
- Task regularity → steady-state scheduling.
- Assessing centralized versus decentralized approaches
Large-scale distributed platforms result from the collaboration of many users:

▶ sharing resources amongst users should somehow be fair.
▶ Task regularity $\leadsto$ steady-state scheduling.
▶ Assessing centralized versus decentralized approaches
Large-scale distributed platforms result from the collaboration of many users:

- sharing resources amongst users should somehow be fair.
- Task regularity $\leadsto$ steady-state scheduling.
- Assessing centralized versus decentralized approaches
Introduction – Applications

- Multiple applications:
  - each consisting in a large number of same-size independent tasks
  - all competing for CPU and network resources

- Different communication and computation demands for different applications

- Important parameter: communication size
  - computation size
Introduction – Applications

- Multiple applications:
  - each consisting in a large number of same-size independent tasks
  - all competing for CPU and network resources

- Different communication and computation demands for different applications

- Important parameter: communication size, computation size
Introduction – Applications

- Multiple applications:
  - each consisting in a large number of same-size independent tasks
  - all competing for CPU and network resources

- Different communication and computation demands for different applications

- Important parameter: 
  \[
  \text{communication size} \\
  \text{computation size}
  \]
Target platform: master-worker
star network

Master holds all tasks initially
Maximize **throughput** (number of tasks processed per unit of time).

Maintain balanced execution between applications (**fairness**)

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

**Objective function:**
- priority weight: \( w^{(k)} \) for application \( A_k \)
- throughput:
  \[ \alpha^{(k)} = \text{number of tasks of type } k \text{ computed per time-unit} \]
- MAX-MIN fairness: MAXIMIZE \( \min_k \left\{ \frac{\alpha^{(k)}}{w^{(k)}} \right\} \).
Maximize **throughput** (number of tasks processed per unit of time).

Maintain balanced execution between applications (**fairness**).

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

Objective function:
- priority weight: \( w^{(k)} \) for application \( A_k \)
- throughput:
  \[ \alpha^{(k)} = \text{number of tasks of type } k \text{ computed per time-unit} \]
- MAX-MIN fairness: \( \text{MAXIMIZE } \min_k \left\{ \frac{\alpha^{(k)}}{w^{(k)}} \right\} \).
Maximize throughput (number of tasks processed per unit of time).

Maintain balanced execution between applications (fairness).

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

Objective function:
- priority weight: \( w^{(k)} \) for application \( A_k \)
- throughput: \( \alpha^{(k)} = \text{number of tasks of type } k \text{ computed per time-unit} \)
- MAX-MIN fairness: \( \text{MAXIMIZE} \min_k \left\{ \frac{\alpha^{(k)}}{w^{(k)}} \right\} \).
Introduction – Goals

- Maximize **throughput** (number of tasks processed per unit of time).
- Maintain balanced execution between applications (*fairness*).
- Scheduling decisions:
  - at master: which applications to assign to which subtree
  - at nodes (tree): which tasks to forward to which children
- Objective function:
  - priority weight: $w^{(k)}$ for application $A_k$
  - throughput: $\alpha^{(k)} = \text{number of tasks of type } k \text{ computed per time-unit}$
  - MAX-MIN fairness: $\text{MAXIMIZE } \min_k \left\{ \frac{\alpha^{(k)}}{w^{(k)}} \right\}$.
Outline

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
  - Workers $P_1, \ldots, P_p$, master $P_{\text{master}}$
  - Parent of $P_u$: $P_{p(u)}$
  - Bandwidth of link $P_{p(u)} \rightarrow P_u$: $b_u$
  - Computing speed of $P_u$: $c_u$
  - Full communication/computation overlap
  - One-port model
Platform Model

- Star or tree network
- Workers $P_1, \ldots, P_p$, master $P_{\text{master}}$
- Parent of $P_u$: $P_{p(u)}$
- Bandwidth of link $P_{p(u)} \rightarrow P_u$: $b_u$
- Computing speed of $P_u$: $c_u$
- Full communication/computation overlap
- One-port model
Platform and Application Model

Platform Model

- Star or tree network
- Workers $P_1, \ldots, P_p$, master $P_{\text{master}}$
- Parent of $P_u$: $P_{p(u)}$
- Bandwidth of link $P_{p(u)} \rightarrow P_u$: $b_u$
- Computing speed of $P_u$: $c_u$
- Full communication/computation overlap
- One-port model
Platform Model

- Star or tree network
- Workers $P_1, \ldots, P_p$, master $P_{\text{master}}$
- Parent of $P_u$: $P_{p(u)}$
- Bandwidth of link $P_{p(u)} \rightarrow P_u$: $b_u$
- Computing speed of $P_u$: $c_u$
- Full communication/computation overlap
- One-port model
Platform and Application Model

Platform Model

- Star or tree network
- Workers $P_1, \ldots, P_p$, master $P_{\text{master}}$
- Parent of $P_u$: $P_{p(u)}$
- Bandwidth of link $P_{p(u)} \rightarrow P_u$: $b_u$
- Computing speed of $P_u$: $c_u$
- Full communication/computation overlap
- One-port model
Platform Model

- Star or tree network
- Workers $P_1, \ldots, P_p$, master $P_{\text{master}}$
- Parent of $P_u$: $P_{p(u)}$
- Bandwidth of link $P_{p(u)} \rightarrow P_u$: $b_u$
- Computing speed of $P_u$: $c_u$
- Full communication/computation overlap
- One-port model
Platform Model

- Star or tree network
- Workers $P_1, \ldots, P_p$, master $P_{\text{master}}$
- Parent of $P_u$: $P_{p(u)}$
- Bandwidth of link $P_{p(u)} \rightarrow P_u$: $b_u$
- Computing speed of $P_u$: $c_u$
- Full communication/computation overlap
- One-port model
Platform and Application Model

Application Model

- \( K \) applications \( A_1, \ldots, A_k \)

- Priority weights \( w^{(k)}: w^{(1)} = 3 \) and \( w^{(2)} = 1 \)

  \[ \iff \text{ process 3 tasks of type 1 per task of type 2} \]

- For each task of \( A_k \):
  - processing cost \( c^{(k)} \) (MFlops)
  - communication cost \( b^{(k)} \) (MBytes)

- Communication for input 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 $w^{(2)} = 1$  
  $\iff$ process 3 tasks of type 1 per task of type 2
- For each task of $A_k$:
  - processing cost $c^{(k)}$ (MFlops)
  - communication cost $b^{(k)}$ (MBytes)
- Communication for input 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 \( w^{(2)} = 1 \)
  \( \iff \) process 3 tasks of type 1 per task of type 2
- For each task of \( A_k \):
  - processing cost \( c^{(k)} \) (MFlops)
  - communication cost \( b^{(k)} \) (MBytes)
- Communication for input 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 $w^{(2)} = 1$
  $\iff$ process 3 tasks of type 1 per task of type 2
- For each task of $A_k$:
  - processing cost $c^{(k)}$ (MFlops)
  - communication cost $b^{(k)}$ (MBytes)
- Communication for input data only (no result message)
- communication-to-computation ratio (CCR): $\frac{b^{(k)}}{c^{(k)}}$
Platform and Application Model

Application Model

- $K$ applications $A_1, \ldots, A_k$
- Priority weights $w^{(k)}$: $w^{(1)} = 3$ and $w^{(2)} = 1$
  \[ \iff \text{process 3 tasks of type 1 per task of type 2} \]
- For each task of $A_k$:
  - processing cost $c^{(k)}$ (MFlops)
  - communication cost $b^{(k)}$ (MBytes)
- Communication for input data only (no result message)
- communication-to-computation ratio (CCR): $\frac{b^{(k)}}{c^{(k)}}$
Outline

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

**Background**  Approach pioneered by Bertsimas and Gamarnik

**Rationale**  Maximize throughput (total load executed per period)

**Simplicity**  Relaxation of makespan minimization problem
- Ignore initialization and clean-up phases
- Precise ordering of tasks/messages not needed
- Characterize resource activity during each time-unit:
  - which (rational) fraction of time is spent computing for which application?
  - which (rational) fraction of time is spent receiving or sending to which neighbor?
Steady-state Scheduling

**Background**  Approach pioneered by Bertsimas and Gamarnik

**Rationale**  Maximize throughput (total load executed per period)

- Simplicity  Relaxation of makespan minimization problem

- Ignore initialization and clean-up phases
- Precise ordering of tasks/messages not needed
- Characterize resource activity during each time-unit:
  - which (rational) fraction of time is spent computing for which application?
  - which (rational) fraction of time is spent receiving or sending to which neighbor?
Computing the Optimal Solution

Steady-state Scheduling

Background  Approach pioneered by Bertsimas and Gamarnik

Rationale  Maximize throughput (total load executed per period)

Simplicity  Relaxation of makespan minimization problem

- Ignore initialization and clean-up phases
- Precise ordering of tasks/messages not needed
- Characterize resource activity during each time-unit:
  - which (rational) fraction of time is spent computing for which application?
  - which (rational) fraction of time is spent receiving or sending to which neighbor?
Steady-state Scheduling

**Background**  Approach pioneered by Bertsimas and Gamarnik

**Rationale**  Maximize throughput (total load executed per period)

**Simplicity**  Relaxation of makespan minimization problem
- Ignore initialization and clean-up phases
- Precise ordering of tasks/messages not needed
- Characterize resource activity during each time-unit:
  - which (rational) fraction of time is spent computing for which application?
  - which (rational) fraction of time is spent receiving or sending to which neighbor?
Steady-state Scheduling

**Background**  
Approach pioneered by Bertsimas and Gamarnik

**Rationale**  
Maximize throughput (total load executed per period)

**Simplicity**  
Relaxation of makespan minimization problem

- Ignore initialization and clean-up phases
- Precise ordering of tasks/messages not needed
- Characterize resource activity during each time-unit:
  - which (rational) fraction of time is spent computing for which application?
  - which (rational) fraction of time is spent receiving or sending to which neighbor?
Steady-state Scheduling

**Background** Approach pioneered by Bertsimas and Gamarnik

**Rationale** Maximize throughput (total load executed per period)

**Simplicity** Relaxation of makespan minimization problem

- Ignore initialization and clean-up phases
- Precise ordering of tasks/messages not needed
- Characterize resource activity during each time-unit:
  - which (rational) fraction of time is spent computing for which application?
  - which (rational) fraction of time is spent receiving or sending to which neighbor?
Computing the Optimal Solution

Linear Program for a Star Network

- $\alpha_u^{(k)} =$ rational number of tasks of $A_k$ executed by $P_u$ every time-unit
- Throughput for application $A_k$: $\alpha^{(k)} = \sum_{u=1}^{p} \alpha_u^{(k)}$
- Objective: MAXIMIZE $\min_k \frac{\alpha^{(k)}}{w^{(k)}}$
- Constraint for computations by $P_u$: $\sum_k \alpha_u^{(k)} \cdot c^{(k)} \leq c_u$
- Number of bytes sent to worker $P_u$: $\sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)}$
- Constraint for communications from the master: $\sum_{u=1}^{p} \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)} \leq b_u$
- The previous linear program can be solved in polynomial time.
Computing the Optimal Solution

Linear Program for a Star Network

- $\alpha^{(k)}_u = \text{rational number of tasks of } A_k \text{ executed by } P_u \text{ every time-unit}$
- Throughput for application $A_k$: $\alpha^{(k)} = \sum_{u=1}^{p} \alpha^{(k)}_u$
- Objective: MAXIMIZE $\min_k \frac{\alpha^{(k)}}{w^{(k)}}$
- Constraint for computations by $P_u$: $\sum_k \alpha^{(k)}_u \cdot c^{(k)} \leq c_u$
- Number of bytes sent to worker $P_u$: $\sum_{k=1}^{K} \alpha^{(k)}_u \cdot b^{(k)}$
- Constraint for communications from the master: $\sum_{u=1}^{p} \sum_{k=1}^{K} \alpha^{(k)}_u \cdot b^{(k)} \leq b_u$
- The previous linear program can be solved in polynomial time.
Computing the Optimal Solution

Linear Program for a Star Network

- $\alpha_u^{(k)}$ = rational number of tasks of $A_k$ executed by $P_u$ every time-unit
- Throughput for application $A_k$: $\alpha^{(k)} = \sum_{u=1}^{p} \alpha_u^{(k)}$
- Objective: MAXIMIZE $\min_k \frac{\alpha^{(k)}}{w^{(k)}}$
- Constraint for computations by $P_u$: $\sum_k \alpha_u^{(k)} \cdot c^{(k)} \leq c_u$
- Number of bytes sent to worker $P_u$: $\sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)}$
- Constraint for communications from the master: $\sum_{u=1}^{p} \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)} \leq b_u$
- The previous linear program can be solved in polynomial time.
Computing the Optimal Solution

Linear Program for a Star Network

- $\alpha_u^{(k)}$ = rational number of tasks of $A_k$ executed by $P_u$ every time-unit
- Throughput for application $A_k$: $\alpha^{(k)} = \sum_{u=1}^{p} \alpha_u^{(k)}$
- Objective: MAXIMIZE $\min_k \frac{\alpha^{(k)}}{w^{(k)}}$
- Constraint for computations by $P_u$:
  \[ \sum_k \alpha_u^{(k)} \cdot c^{(k)} \leq c_u \]
- Number of bytes sent to worker $P_u$: $\sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)}$
- Constraint for communications from the master:
  \[ \sum_{u=1}^{p} \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)} \leq b_u \]
- The previous linear program can be solved in polynomial time.
Computing the Optimal Solution

Linear Program for a Star Network

- \( \alpha^{(k)}_u \) = rational number of tasks of \( A_k \) executed by \( P_u \) every time-unit
- Throughput for application \( A_k \): \( \alpha^{(k)} = \sum_{u=1}^{p} \alpha^{(k)}_u \)
- Objective: MAXIMIZE \( \min_k \frac{\alpha^{(k)}}{w^{(k)}} \)
- Constraint for computations by \( P_u \):
  \[ \sum_k \alpha^{(k)}_u \cdot c^{(k)} \leq c_u \]
- Number of bytes sent to worker \( P_u \): \( \sum_{k=1}^{K} \alpha^{(k)}_u \cdot b^{(k)} \)
- Constraint for communications from the master:
  \[ \sum_{u=1}^{p} \sum_{k=1}^{K} \alpha^{(k)}_u \cdot b^{(k)} \leq b_u \]
- The previous linear program can be solved in polynomial time.
Computing the Optimal Solution

Linear Program for a Star Network

- $\alpha_u^{(k)}$ = rational number of tasks of $A_k$ executed by $P_u$ every time-unit.
- Throughput for application $A_k$: $\alpha^{(k)} = \sum_{u=1}^{p} \alpha_u^{(k)}$
- Objective: MAXIMIZE $\min_k \frac{\alpha^{(k)}}{w^{(k)}}$
- Constraint for computations by $P_u$: 
  \[ \sum_k \alpha_u^{(k)} \cdot c^{(k)} \leq c_u \]
- Number of bytes sent to worker $P_u$: 
  \[ \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)} \]
- Constraint for communications from the master:
  \[ \sum_{u=1}^{p} \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)} \leq b_u \]
- The previous linear program can be solved in polynomial time.
Computing the Optimal Solution

Linear Program for a Star Network

- $\alpha_u^{(k)}$ = rational number of tasks of $A_k$ executed by $P_u$ every time-unit
- Throughput for application $A_k$: $\alpha^{(k)} = \sum_{u=1}^{p} \alpha_u^{(k)}$
- Objective: MAXIMIZE $\min_k \frac{\alpha^{(k)}}{w^{(k)}}$
- Constraint for computations by $P_u$: $\sum_k \alpha_u^{(k)} \cdot c^{(k)} \leq c_u$
- Number of bytes sent to worker $P_u$: $\sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)}$
- Constraint for communications from the master: $\sum_{u=1}^{p} \sum_{k=1}^{K} \alpha_u^{(k)} \cdot b^{(k)} \leq b_u$
- The previous linear program can be solved in polynomial time.
Reconstructing an Optimal Schedule

- Solution of linear program: $\alpha^{(k)}_u = \frac{p_{u,k}}{q_{u,k}}$, throughput $\rho$
- Set period length: $T_p = \text{lcm}\{q_{u,k}\}$
- During 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 optimal number of tasks in time $T$, up to a constant independent of $T$)
Computing the Optimal Solution

Reconstructing an Optimal Schedule

- Solution of linear program: \( \alpha_u^{(k)} = \frac{p_{u,k}}{q_{u,k}} \), throughput \( \rho \)
- Set period length: \( T_p = \text{lcm}\{q_{u,k}\} \)
- During 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 optimal number of tasks in time \( T \), up to a constant independent of \( T \))
Computing the Optimal Solution

Reconstructing an Optimal Schedule

- Solution of linear program: \( \alpha_u^{(k)} = \frac{p_{u,k}}{q_{u,k}} \), throughput \( \rho \)
- Set period length: \( T_p = \text{lcm}\{q_{u,k}\} \)
- During 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 optimal number of tasks in time \( T \), up to a constant independent of \( T \))
Computing the Optimal Solution

Reconstructing an Optimal Schedule

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

Reconstructing an Optimal Schedule

- Solution of linear program: \( \alpha_u^{(k)} = \frac{p_{u,k}}{q_{u,k}} \), throughput \( \rho \)
- Set period length: \( T_p = \text{lcm}\{q_{u,k}\} \)
- During 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 optimal number of tasks in time \( T \), up to a constant independent of \( T \))
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 extended
- Similar reconstruction of periodic schedule
- No proof of a particular structure
Adaptation to Tree Networks

- Linear Program can be extended
- Similar reconstruction of periodic schedule
- No proof of a particular structure
Adaptation to Tree Networks

- Linear Program can be extended
- Similar reconstruction of periodic schedule
- No proof of a particular structure
Computing the Optimal Solution

Problems in previous solutions

- LP approach:
  - Centralized, needs all global information at master
  - Schedule period possibly huge
    - difficult to adapt to load variation
  - Large memory requirement, huge flow time
Problems in previous solutions

- LP approach:
  - Centralized, needs all global information at master
  - Schedule period possibly huge
    \(\rightarrow\) difficult to adapt to load variation
  - Large memory requirement, huge flow time
Problems in previous solutions

- LP approach:
  - Centralized, needs all global information at master
  - Schedule period possibly huge
    - difficult to adapt to load variation
  - Large memory requirement, huge flow time
Outline

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

- 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

Arnaud Legrand
Scheduling multiple bag-of-task applications 18/33
Centralized LP based (LP)
- Solve linear program with global information
- Give each node the $\alpha^{(k)}_u$ for its children and himself
- Use a 1D load balancing mechanism with these ratios → close to optimal throughput?
- Hybrid heuristic: **centralized** computation of rates ($\alpha^{(k)}_u$) but **distributed** control of the scheduling

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 $\alpha^{(k)}_u$ for its children and himself
  - Use a 1D load balancing mechanism with these ratios → close to optimal throughput?
  - Hybrid heuristic: **centralized** computation of rates $(\alpha^{(k)}_u)$ but **distributed** control of the scheduling

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

Heuristics – One application = bandwidth-centric strategy

▶ Optimal strategy for a single application: send tasks to faster-communicating children first

▶ Demand-driven based on local information: bandwidth and CPU speed of children

▶ Extension to trees by bottom-up node reduction
Coarse-Grain Bandwidth-Centric (CGBC)

- Bandwidth-centric = optimal solution for a single application (send tasks to children communicating faster first)
- Assemble different types of tasks into one macro-task:

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

- Not expected to reach optimal throughput: slow links are used to transfer tasks with high CCR
Parallel Bandwidth-Centric (PBC)
- Superpose bandwidth-centric strategy for each application
- On each worker, \( K \) independent non-cooperative schedulers
- Independent schedulers → concurrent transfers and concurrent computations
- Limited capacity on outgoing port
- Gives an (unfair) advantage to PBC (allows interruptible communications)
Parallel Bandwidth-Centric (PBC)

- Superpose bandwidth-centric strategy for each application
- On each worker, $K$ independent non-cooperative schedulers
- Independent schedulers $\rightarrow$ concurrent transfers and concurrent computations
- Limited capacity on outgoing port
  $\leadsto$ gives an (unfair) advantage to PBC (allows interruptible communications)
Parallel Bandwidth-Centric (PBC)

- Superpose bandwidth-centric strategy for each application
- On each worker, $K$ independent non-cooperative schedulers
- Independent schedulers $\rightarrow$ concurrent transfers and concurrent computations
- Limited capacity on outgoing port
  $\sim$ gives an (unfair) advantage to PBC (allows interruptible communications)
Decentralized Heuristics

Heuristics – PBC

- Parallel Bandwidth-Centric (PBC)
  - Superpose bandwidth-centric strategy for each application
  - On each worker, $K$ independent non-cooperative schedulers
  - Independent schedulers $\rightarrow$ concurrent transfers and concurrent computations
  - Limited capacity on outgoing port
    $\sim$ gives an (unfair) advantage to PBC (allows interruptible communications)
Decentralized Heuristics

Heuristics – DATA-CENTRIC

- Data-centric scheduling (DATA-CENTRIC)
  - Decentralized heuristic
  - Try to convergence to the solution of LP
    - Intuition based on the structure of optimal solution for star networks
    - 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, in order 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 LP
- Intuition based on the structure of optimal solution for star networks
- 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, in order 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 LP
- Intuition based on the structure of optimal solution for star networks
- 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, in order 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 LP
- Intuition based on the structure of optimal solution for star networks
- 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, in order 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 LP
- Intuition based on the structure of optimal solution for star networks
- 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, in order 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 phase where all applications simultaneously run
    → $T =$ first time s.t. all tasks of some application are terminated
  ▶ Ignore initialization and termination phases
  ▶ Set time-interval: $[0.1 \times T ; \ 0.9 \times T]$
  ▶ Compute achieved throughput for each application on this interval.
    $\rho_{\text{experimental}}$ denotes the smallest throughput.

▶ 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,
  ▶ Link and processor capacities measured within SimGrid
Methodology

- How to measure fair-throughput?
  - Concentrate on phase where all applications simultaneously run
    \( T \) = first time s.t. all tasks of some application are terminated
  - Ignore initialization and termination phases
  - Set time-interval: \([0.1 \times T ; 0.9 \times T]\)
  - Compute achieved throughput for each application on this interval. 
    \( \rho_{\text{experimental}} \) denotes the smallest throughput.

- 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,
  - Link and processor capacities measured within SimGrid
Methodology

- How to measure fair-throughput?
  - Concentrate on phase where all applications simultaneously run
    \( T \) = first time s.t. all tasks of some application are terminated
  - Ignore initialization and termination phases
  - Set time-interval: \([0.1 \times T; 0.9 \times T]\)
  - Compute achieved throughput for each application on this interval.
    \( \rho_{\text{experimental}} \) denotes the smallest throughput.

- 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,
  - Link and processor capacities measured within SimGrid
Methodology

- **How to measure fair-throughput?**
  - Concentrate on phase where all applications simultaneously run
    \[ T = \text{first time s.t. all tasks of some application are terminated} \]
  - Ignore initialization and termination phases
  - Set time-interval: \( [0.1 \times T; 0.9 \times T] \)
  - Compute achieved throughput for each application on this interval.
    \[ \rho_{\text{experimental}} \] denotes the smallest throughput.

- **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,
  - Link and processor capacities measured within SimGrid
Simulation Results

Theoretical v/ Experimental Throughput

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

\[
\text{deviation} = \frac{\rho_{\text{theoretical}} - \rho_{\text{experimental}}}{\rho_{\text{theoretical}}}
\]

average deviation = 9.4%

average deviation independant of heuristic

- Increase buffer size from 10 to 100 \(\rightarrow\) 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
- Plot distribution
Simulation Results

Theoretical v/ Experimental Throughput

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

\[
\text{deviation} = \frac{\rho_{\text{theoretical}} - \rho_{\text{experimental}}}{\rho_{\text{theoretical}}}
\]

average deviation = 9.4%

average deviation independent of heuristic

- Increase buffer size from 10 to 100 \(\sim\) 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

- Plot distribution
Simulation Results

Theoretical v/ Experimental Throughput

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

\[ \text{deviation} = \frac{\rho_{\text{theoretical}} - \rho_{\text{experimental}}}{\rho_{\text{theoretical}}} \]

average deviation = 9.4%

average deviation independent of heuristic

- Increase buffer size from 10 to 100 \(\Rightarrow\) average deviation = 0.3%

- In the following, LP = basis for comparison

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

- Plot distribution
Simulation Results

Theoretical v/ Experimental Throughput

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

\[
\text{deviation} = \frac{\rho_{\text{theoretical}} - \rho_{\text{experimental}}}{\rho_{\text{theoretical}}}
\]

average deviation = 9.4%

average deviation independent of heuristic

- Increase buffer size from 10 to 100 \(\sim\) 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

- Plot distribution
Simulation Results

Theoretical v/ Experimental Throughput

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

\[ \text{deviation} = \frac{\rho_{\text{theoretical}} - \rho_{\text{experimental}}}{\rho_{\text{theoretical}}} \]

average deviation = 9.4%

average deviation independent of heuristic

- Increase buffer size from 10 to 100 \( \sim \) 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

- Plot distribution
Theoretical v/ Experimental Throughput

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

\[
\text{deviation} = \frac{\rho_{\text{theoretical}} - \rho_{\text{experimental}}}{\rho_{\text{theoretical}}}
\]

average deviation = 9.4%

average deviation independent of heuristic

- Increase buffer size from 10 to 100 \(\sim\) 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
- Plot distribution
Simulation Results

Theoretical v/ Experimental Throughput

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

\[
\text{deviation} = \frac{\rho_{\text{theoretical}} - \rho_{\text{experimental}}}{\rho_{\text{theoretical}}}
\]

Average deviation = 9.4%

Average deviation independent of heuristic

- Increase buffer size from 10 to 100 \(\leadsto\) 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

- Plot distribution
Performance of FCFS

- Geometrical average: FCFS is 1.56 times worse than LP
- Worst case: 8 times worse
Performance of CGBC

- 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.
Simulation Results

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 badly in few cases
Outline

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

- Centralized algorithm computes optimal solution with global information
- Nice characterization of optimal solution on single-level trees
- Design distributed heuristics to deal with practical settings of clusters and grids (distributed information, variability, limited memory)
- Evaluation of these heuristics through extensive simulations
- Good performance of sophisticated heuristics compared to the optimal scheduling
Adapt decentralized MultiCommodity Flow algorithm (Awerbuch & Leighton) to our problem
- Decentralized approach to compute optimal throughput
- Slow convergence speed

Consider other kinds of fairness such as proportional fairness:
- Reasonable: better use of resources than max-min, many other protocol have a similar behavior (e.g. last versions of of TCP)
- Centralized: easy (semi-definite programming instead of LP)
- Decentralized: ??

Study robustness and adaptability of these heuristics...