Job Communication Characterization and its Impact on Meta-scheduling Co-allocated Jobs in a Mini-grid

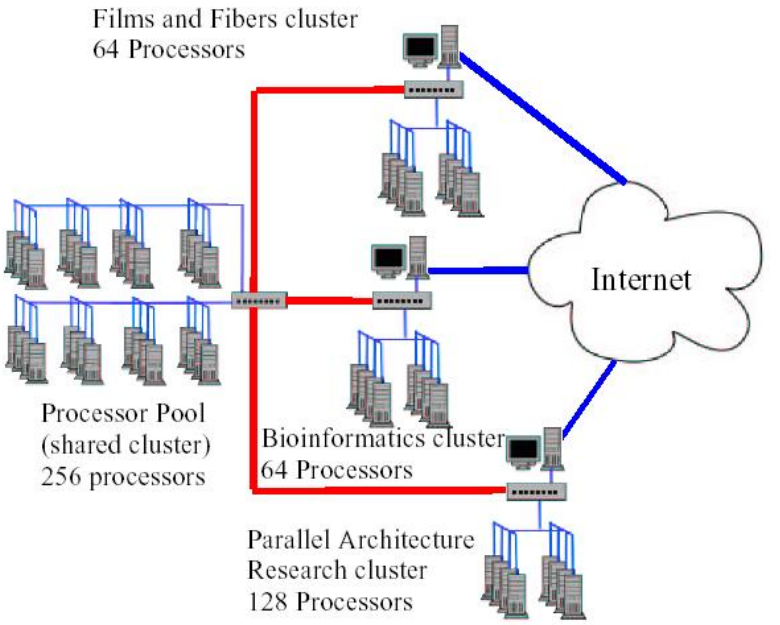
William M. Jones
Louis W. Pang
Dan Stanzione
Walter B. Ligon III
wjones@parl.clemson.edu

Parallel Architecture Research Laboratory
Clemson University
Clemson, South Carolina
http://www.parl.clemson.edu

What is a Mini-grid ?

PARL
Parallel Architecture
Research Laboratory

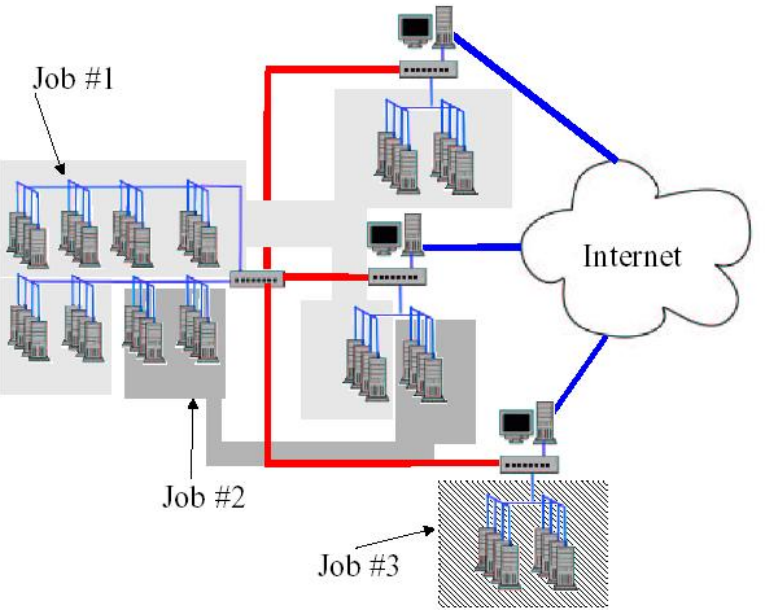
- Geographic co-location
- Dedicated network
- Fine-grain control
- Share resources



Job Scheduling

PARL
Parallel Architecture
Research Laboratory

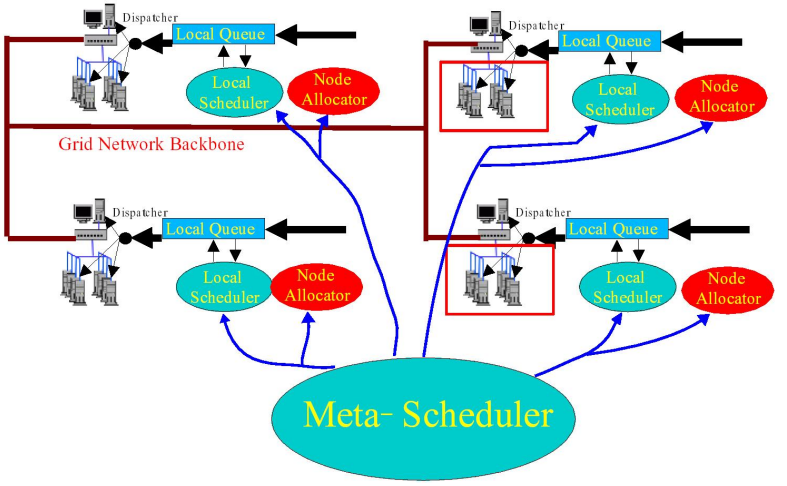
- Local execution
- Job migration
- Job co-allocation
  - Map across boundaries
  - Sharing resources
  - Network BW contention



Initial Goals

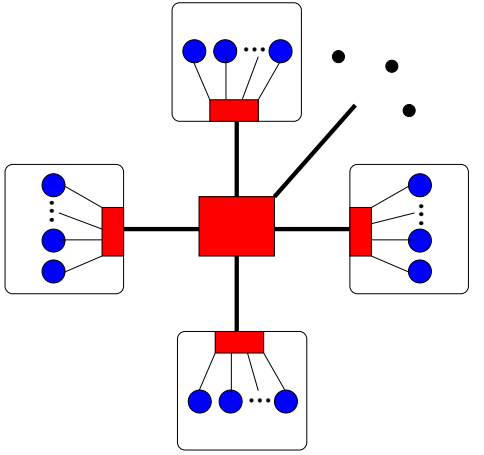
PARL
Parallel Architecture
Research Laboratory

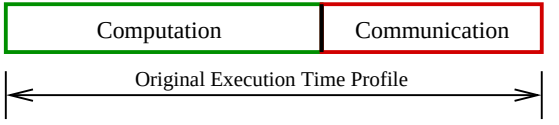
- Meta-scheduler
- Co-allocation
- Create model
- Capture interaction
- Shared network
- Compare and contrast
- Improved scheduling



Models

- Mini-grid
  - Homogeneous nodes
  - Interconnection
- Parallel job
  - Comp. + Comm
  - Network congestion
- Communication
  - All-to-all
  - Per-processor BW



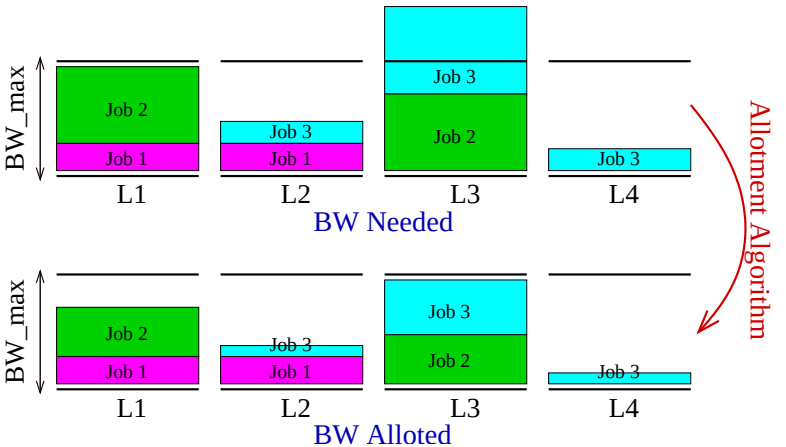


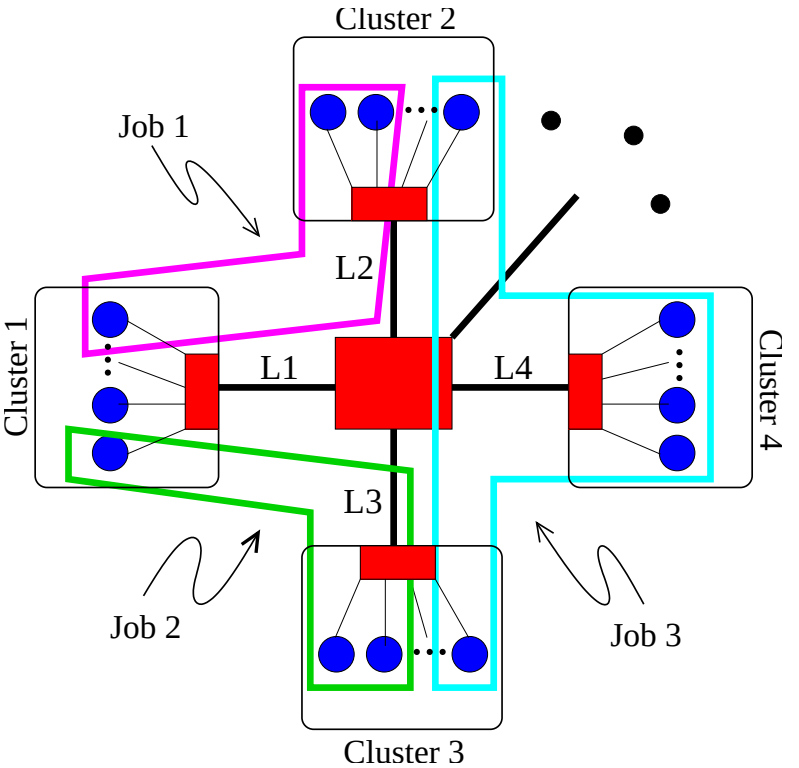
$$BW_i^j = \left(\frac{n_T^j - n_i^j}{n_T^j - 1}\right)(PPBW_j * n_i^j)$$

Job Co-allocation

PARL
Parallel Architecture Research Laboratory

- Bandwidth saturation
  $BW_i^{sat} = BW_i^{max} / \sum_{\forall j \in J_i} BW_i^j$
- Communication slowdown
  $BW_{(k,j)}^{sd} = BW_{(k,j)}^{alloted} / BW_k^j$

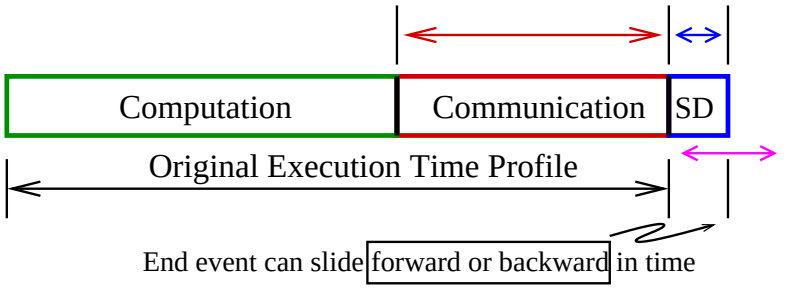




Job Co-allocation Continued

PARL
Parallel Architecture Research Laboratory

- State changing events
  - New co-allc'ed job
  - Co-allc'ed job finishes
- Residual execution times



$$T_E^R = \overbrace{T_C^R + T_P^R}^{\text{Residual Execution Time}}$$

$$T_C^R = \overbrace{(T_C^{R'} - T_C^{\Delta})(BW_{(k,j)}^{sd'})(BW_{(k,j)}^{sd})^{-1}}^{\text{Residual Communication Time}}$$

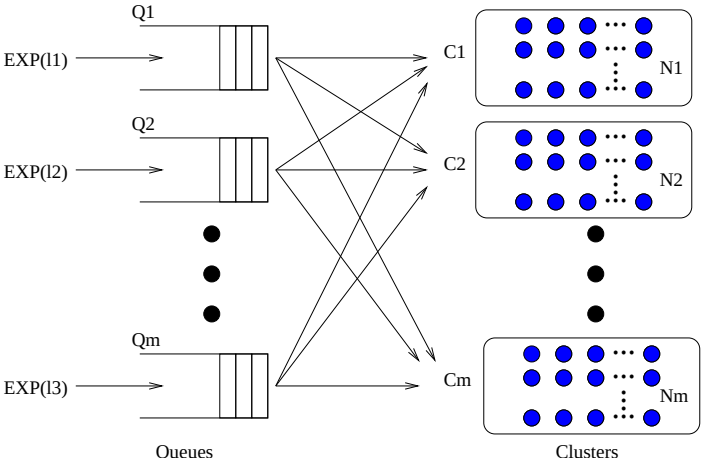
$$T_P^R = \overbrace{(T_P^{R'} - T_P^{\Delta})}^{\text{Residual Computation Time}}$$

$$T_P^{\Delta} = \frac{\Delta T}{T_E^{R'}} T_P^{R'} \quad \text{and} \quad T_C^{\Delta} = \frac{\Delta T}{T_E^{R'}} T_C^{R'}$$

Experimental Setup

PARL
Parallel Architecture
Research Laboratory

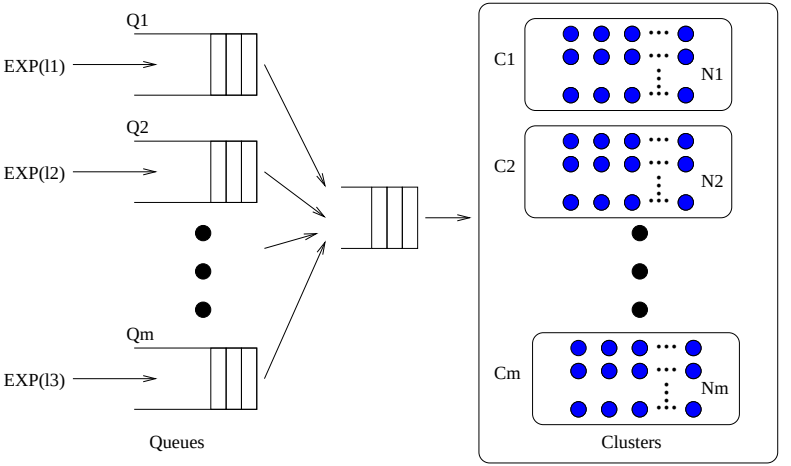
- Synthetic workload
- Poisson AP
- Exp. ST (*initially*)
- 70% comp. / 30% comm.
- 1 Gbps IC network links
- 4,000,000 jobs, $UNIF[10, 90]$
- 100 nodes per cluster



Simulation

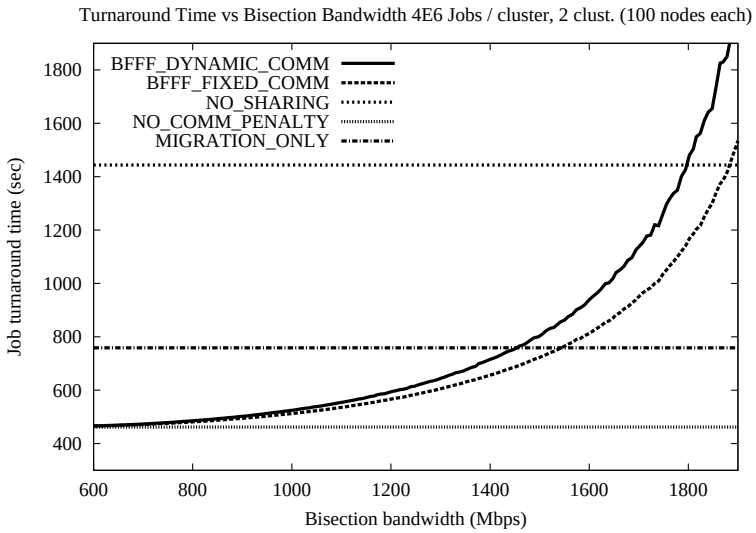
PARL
Parallel Architecture
Research Laboratory

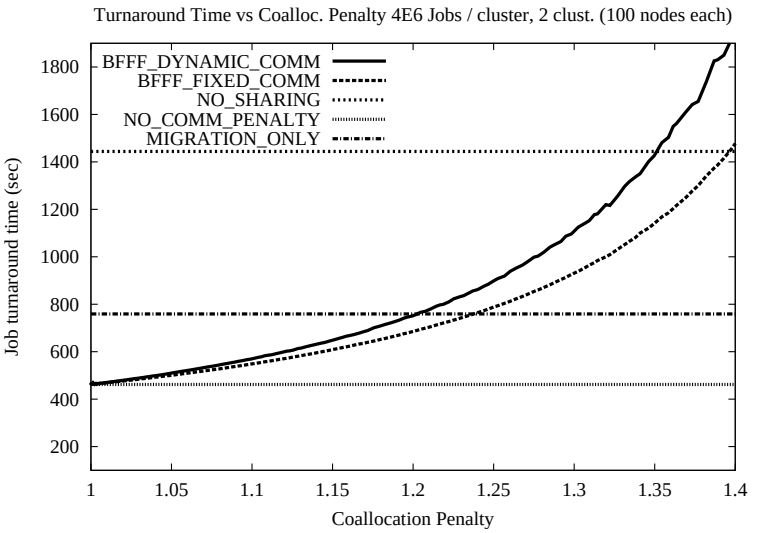
- Algorithms
  - No sharing
  - No comm. penalty (Ideal)
  - Migration only
  - Fixed comm. model
  - Dynamic comm. model
- Policies
  - FCFS Scan
  - Best Fit



Results - 2 Cluster Case

PARL
Parallel Architecture
Research Laboratory

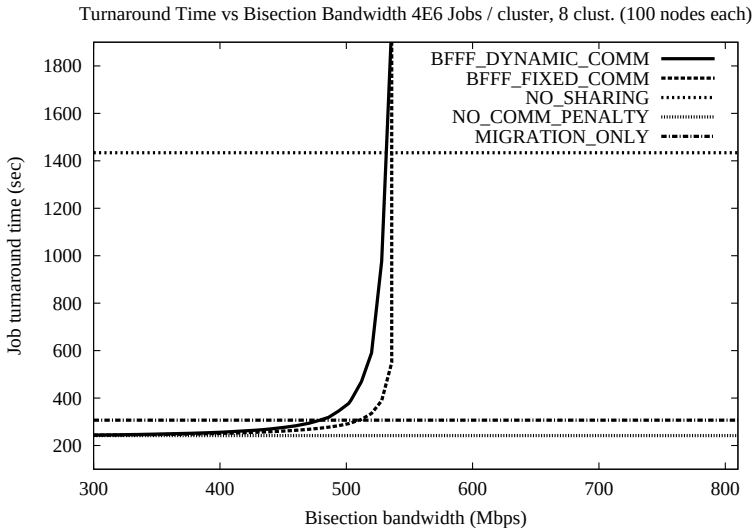


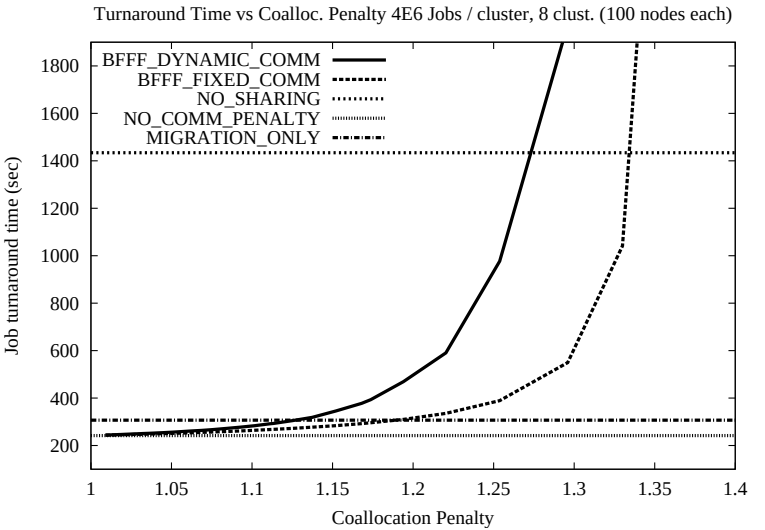


- Dynamic communication model is not as generous as fixed penalty model
- Effects become exacerbated as the number clusters is increased

Results - 8 Cluster Case

PARL
Parallel Architecture
Research Laboratory



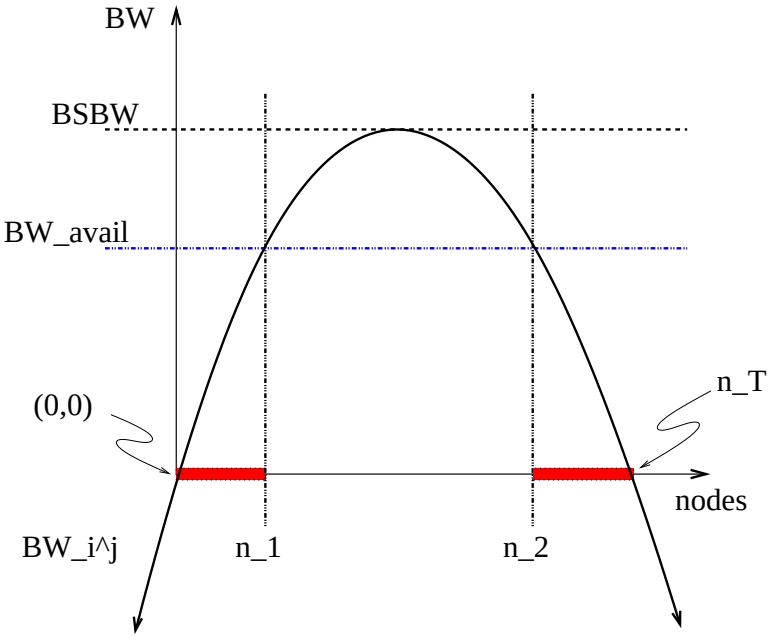


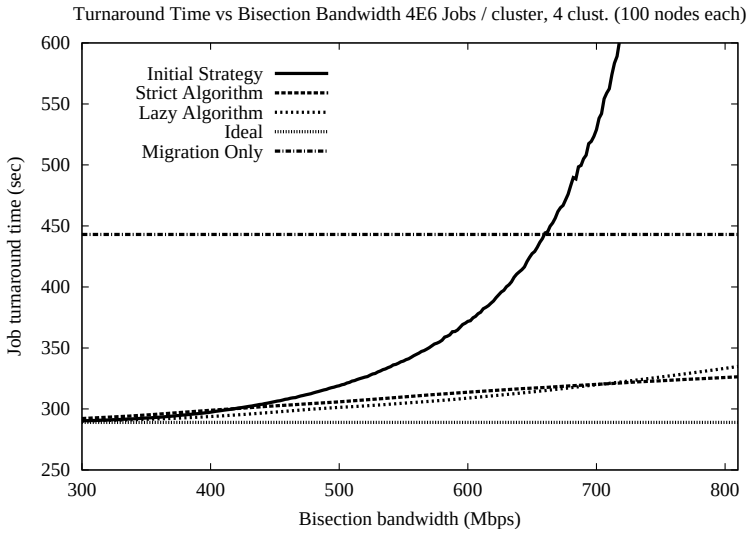
- Dynamic model accounts for the time varying utilization of intercluster NW links
- Captures the interaction of simultaneously co-allocated parallel jobs

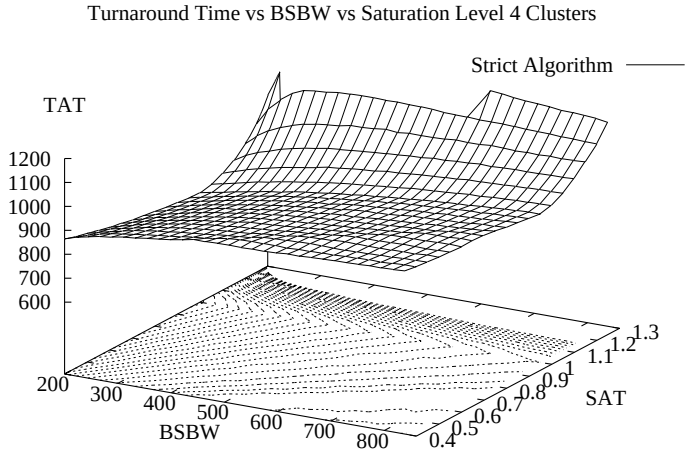
Current Work

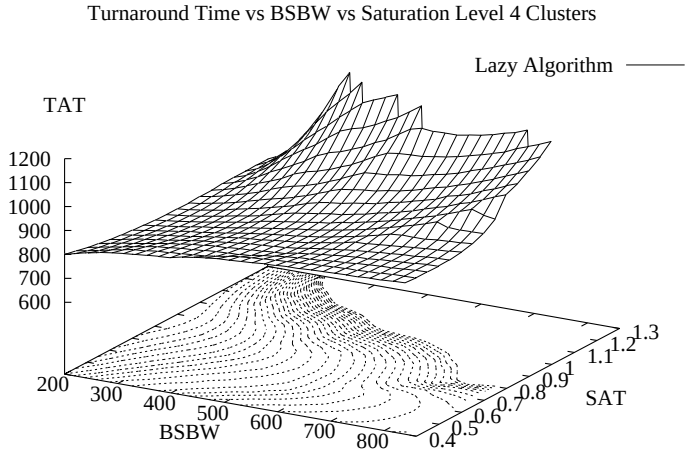
PARL
Parallel Architecture
Research Laboratory

- Constraint satisfaction
- Optimization
- New algorithms
- Foreknowledge



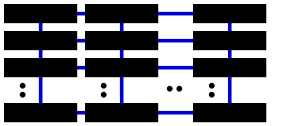

Turnaround Time vs Bisection Bandwidth 4E6 Jobs / cluster, 4 clust. (100 nodes each)
Initial Strategy
Strict Algorithm
Lazy Algorithm
Ideal
Migration Only
Job turnaround time (sec)
Bisection bandwidth (Mbps)
600
550
500
450
400
350
300
250
300
400
500
600
700
800


Turnaround Time vs BSBW vs Saturation Level 4 Clusters
Strict Algorithm
TAT
1200
1100
1000
900
800
700
600
200
300
400
500
600
700
800
BSBW
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
SAT


Turnaround Time vs BSBW vs Saturation Level 4 Clusters
Lazy Algorithm
TAT
1200
1100
1000
900
800
700
600
200
300
400
500
600
700
800
BSBW
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
SAT

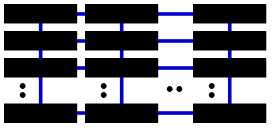
Future Work

PARL
Parallel Architecture
Research Laboratory



- Improve communication model
- Verify saturation model with empirical data
- Extend network to include additional topologies
- Design more *intelligent* scheduling algorithms
- Create more extensive priority mechanisms
- Implement actual scheduler for our mini-grid

END



Questions ?

Bandwidth Allotment Algorithm

PARL
Parallel Architecture Research Laboratory

$$BW_i^{uc\_sat} = \frac{BW_i^{avail}}{\sum_{\forall j \in J_{uc}} BW_{(i,j)}^{alloted}}$$

Step 1: *Initialization* - For every job j, let $BW_{(i,j)}^{alloted} = BW_i^j$. For every link i, let $BW_i^{avail} = BW_i^{max}$. Let the unconstrained set of nodes, J_{uc} = J (set of all jobs). Let the set, J_i be the set of all jobs that span link i.

Step 2: *Saturation detection* - For every link, calculate $BW_i^{uc\_sat}$. While there exists at least one $BW_i^{uc\_sat} < 1.0$, continue, else goto **Step 5**.

Step 3: *Saturation correction* - Identify the link with the smallest $BW_i^{uc\_sat}$ (most saturated link) from **Step 2**, and globally reduce the alloted bandwidth of every job in $J_i \cap J_{uc}$ by a factor of BW_i^{sat}.

Step 4: *Update state* - Remove each of the modified jobs from the set J_{uc}. For each of the modified jobs, remove their alloted bandwidth from the available bandwidth, BW_i^{avail} on each link over which they span. Goto **Step 2**.

Step 5: *Termination* - DONE.