Resource Management and Knapsack Formulations on the Grid

Rafael Parra-Hernandez Daniel C. Vanderster
Nikitas J. Dimopoulos

Department of Computer and Electrical Engineering
University of Victoria, B.C., Canada

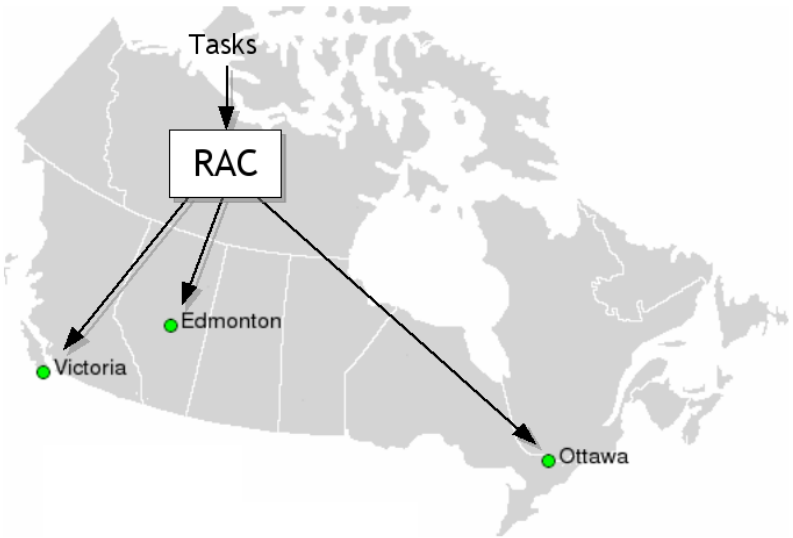
5th IEEE/ACM International Workshop on Grid Computing

University of Victoria

Outline

University of Victoria

Resource Allocation on the Grid



- Grids use a Resource Allocation Centre to make allocation decisions, but:
  - How can we best allocate resources?
  - Can we provide **Quality of Service**?

University of Victoria

Assumptions and Definitions

- Assumptions:
  - We assume a state-based and preemptive form of a Grid.
  - There exists a mechanism which can identify available resources.
  - Task metadata defines each task's resource requirements.
- Definitions:
  - *QoS*: the overall quality of a task's allocation option.
  - *QoS Metric*: a value which characterizes an option.
  - *Utility*: a function which is used to optimize the allocation of resources.

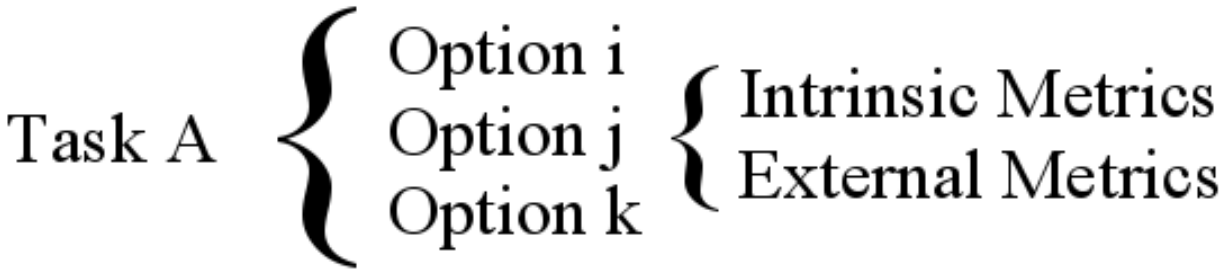
University of Victoria

Outline

University of Victoria

Tasks, Options, and QoS Metrics

- Tasks have a number of options and QoS metrics:



University of Victoria

Example Utility Values

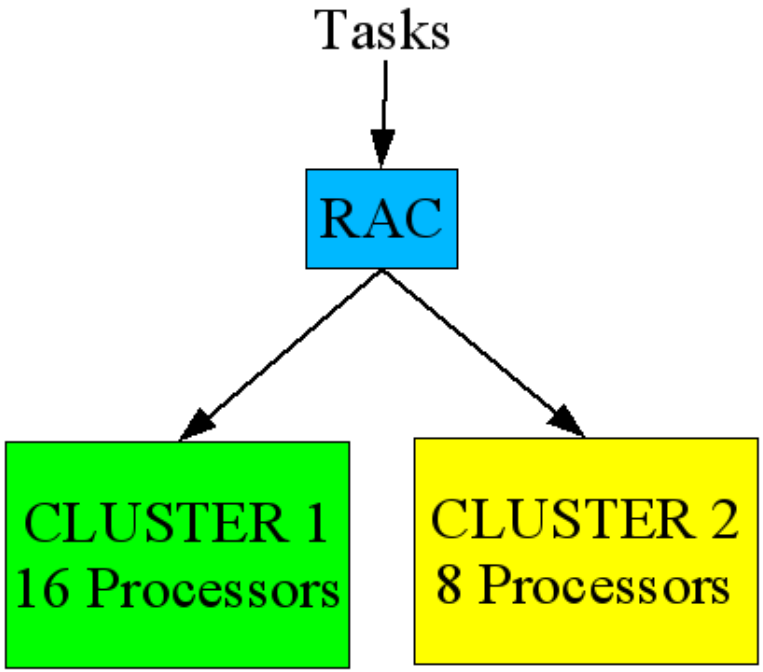
- Ex: A task is submitted with two options:
  1. Use 8 processors and “pay” 100 credits.
  2. Use 1 processor and “pay” 10 credits.
- User-assigned metric must be limited to a maximum overall credit.
  - Ex: A user can “spend” 1000 credits per month.

Outline

University of Victoria

A Small Example Grid

- Assume a Grid with 2 clusters with 1 resource type (processors):



University of Victoria

Two Tasks and Their Options

- Task 1 has three options:
  1. Pay $10 for 8 processors from Cluster 1
  2. Pay $2 for 2 processors from Cluster 1
  3. Pay $9 for 4 and 4 processors from Clusters 1 and 2
- Task 2 has two options:
  1. Pay $11 for 10 processors from Cluster 1
  2. Pay $2 for 1 processor from Cluster 2

Constructing the Problem

- Frame our objective function and constraints:

$$\max f(x) =$$

Subject to

$$\text{Clus. 1:} \quad \leq 16$$

$$\text{Clus. 2:} \quad \leq 8$$

$$= 1$$

$$= 1$$

$$x_{ij} \in \{0, 1\}$$

Constructing the Problem

- Task 1, Option 1: Pay \$10 for 8 processors from Cluster 1:

$$
\begin{aligned}
&\max f(x) = 10x_{11} \\
&\text{Subject to} \\
&\text{Clus. 1:} \quad 8x_{11} \leq 16 \\
&\text{Clus. 2:} \quad \leq 8 \\
&\qquad x_{11} = 1 \\
&\qquad = 1 \\
&\qquad x_{ij} \in \{0, 1\}
\end{aligned}
$$

Constructing the Problem

- Task 1, Option 2: Pay \$2 for 2 processors from Cluster 1:

$$
\begin{aligned}
&\max f(x) = 10x_{11} + 2x_{12} \\
&\text{Subject to} \\
&\text{Clus. 1:} \quad 8x_{11} + 2x_{12} \leq 16 \\
&\text{Clus. 2:} \quad \leq 8 \\
&\qquad x_{11} + x_{12} = 1 \\
&\qquad = 1 \\
&\qquad x_{ij} \in \{0, 1\}
\end{aligned}
$$

Constructing the Problem

- Both tasks and all options filled:

$$
\begin{aligned}
&\max f(x) = 10x_{11} + 2x_{12} + 9x_{13} + 11x_{21} + 2x_{22} \\
&\text{Subject to} \\
&\text{Clus. 1:} \quad 8x_{11} + 2x_{12} + 4x_{13} + 11x_{21} \leq 16 \\
&\text{Clus. 2:} \quad 4x_{13} + 1x_{22} \leq 8 \\
&\qquad x_{11} + x_{12} + x_{13} = 1 \\
&\qquad x_{21} + x_{22} = 1 \\
&\qquad x_{ij} \in \{0, 1\}
\end{aligned}
$$

University of Victoria

The 0,1 Knapsack Problem

- This is a nothing more than a 0,1 Knapsack Problem:

$$\begin{array}{rl} \text{Maximize} & F = \sum_{i=0}^{n} y_i x_i \\ \text{Subject to} & \sum_{i=0}^{n} W_i x_i \leq L \\ & x_i \in \{0, 1\} \end{array}$$

Outline

University of Victoria

Allocation Strategies Used

- We set out to test the knapsack strategy in a large simulated Grid.
- Comparison strategies: FCFS, Backfilling, Gang.
- Knapsack Strategy: Tested a variety of allocation policies. Varied cycling time τ.
- Also tested Knapsack + Backfilling.

Allocation Policy 1: Pure Credit-Value-Driven

- Can we provide QoS, allowing users to prioritize their tasks?
- A user assigns a credit-value v_{ij} to option j of task i.
- In this case, $U(\cdot) = I(\cdot)$, thus:

$$u_{ij} = v_{ij}$$

- The assigned credit-value varies non-linearly with the computation length T_{ij} of the option:

$$\frac{v_{im}}{v_{in}} = \left(\frac{T_{in}}{T_{im}}\right)^{1.1}$$

Allocation Policy 2: Credit-Value-Driven Mediated by Estimated Response Time

- We want to prefer tasks with a long response time.
- Use a credit-value v_{ij} with normalized estimated completion time c_{ij}.

$$u_{ij} = v_{ij} \frac{t - s_i + r_{ij}}{N_i}$$

- t is current time, s_i is submission time, r_{ij} is the remaining computation time.
- N_i is the run-time of the task on a reference processor.

University of Victoria

Allocation Policy 3: Policy 2 with Sigmoidal Utility Function

- We may want to limit the utility function in policy 2.
- Instead of an identity utility function we use a sigmoidal function:

$$U(x) = \frac{2.0}{\exp(-0.5x) + 1} - 1$$

University of Victoria

Allocation Policy 4,5: Estimated Response Time + Closeness to Completion

- Policy 4: Tasks which are very close to completing are given a higher priority:

$$u_{ij} = \frac{t - s_i + r_{ij}}{N_i} + \frac{N_i}{r_{ij}}$$

- Policy 5: uses the sigmoidal utility function shown previously.

University of Victoria

Grid and Task Model

- Simulated Grid has four clusters with 4, 8, 16, and 32 processors.
  - Cluster processors are homogeneous.
- Task sequences are poisson with mean interarrival time λ:
  - Computation demand is bimodal.
  - Varied number of processors and credit-value.
  - Task sequences of length 2000.

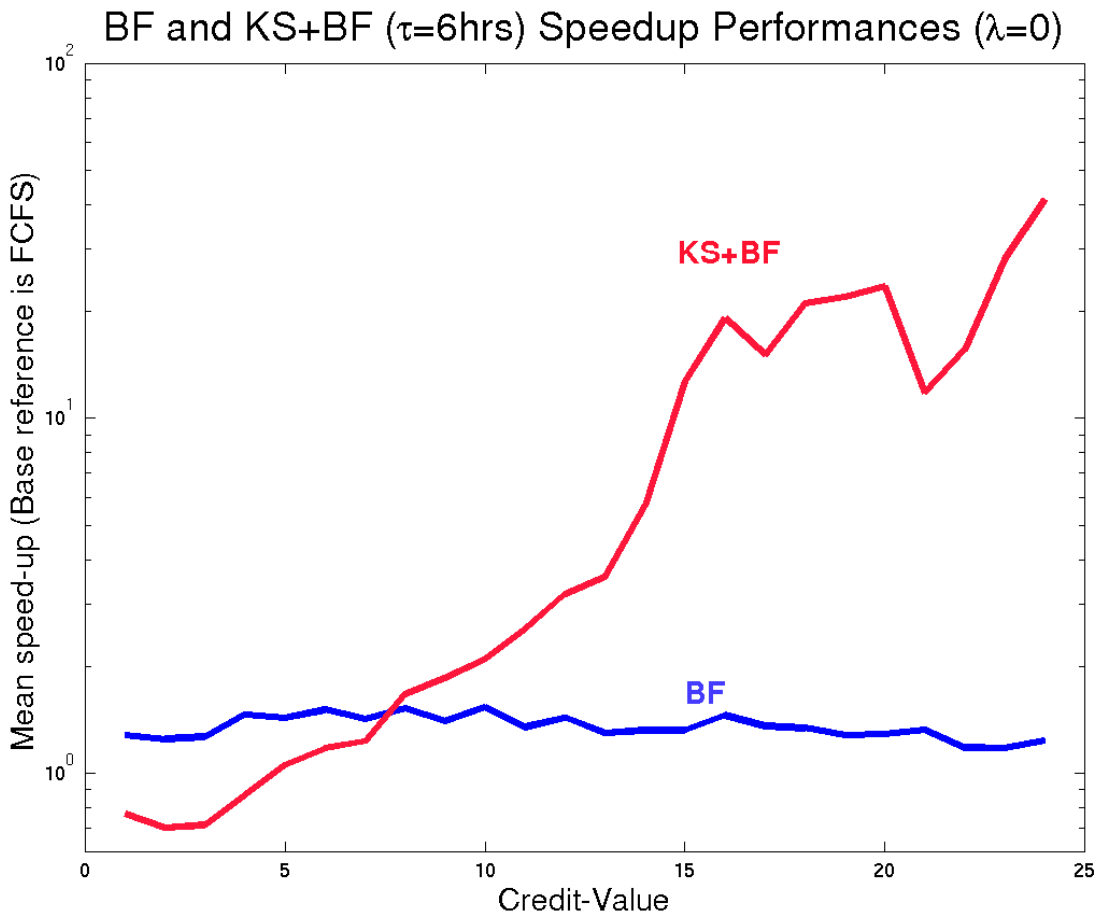
Outline

University of Victoria

Performance Measurements

- Remember, the purpose of our method is to provide QoS without increasing overall completion times.
- As we'll see, the knapsack strategy does well.

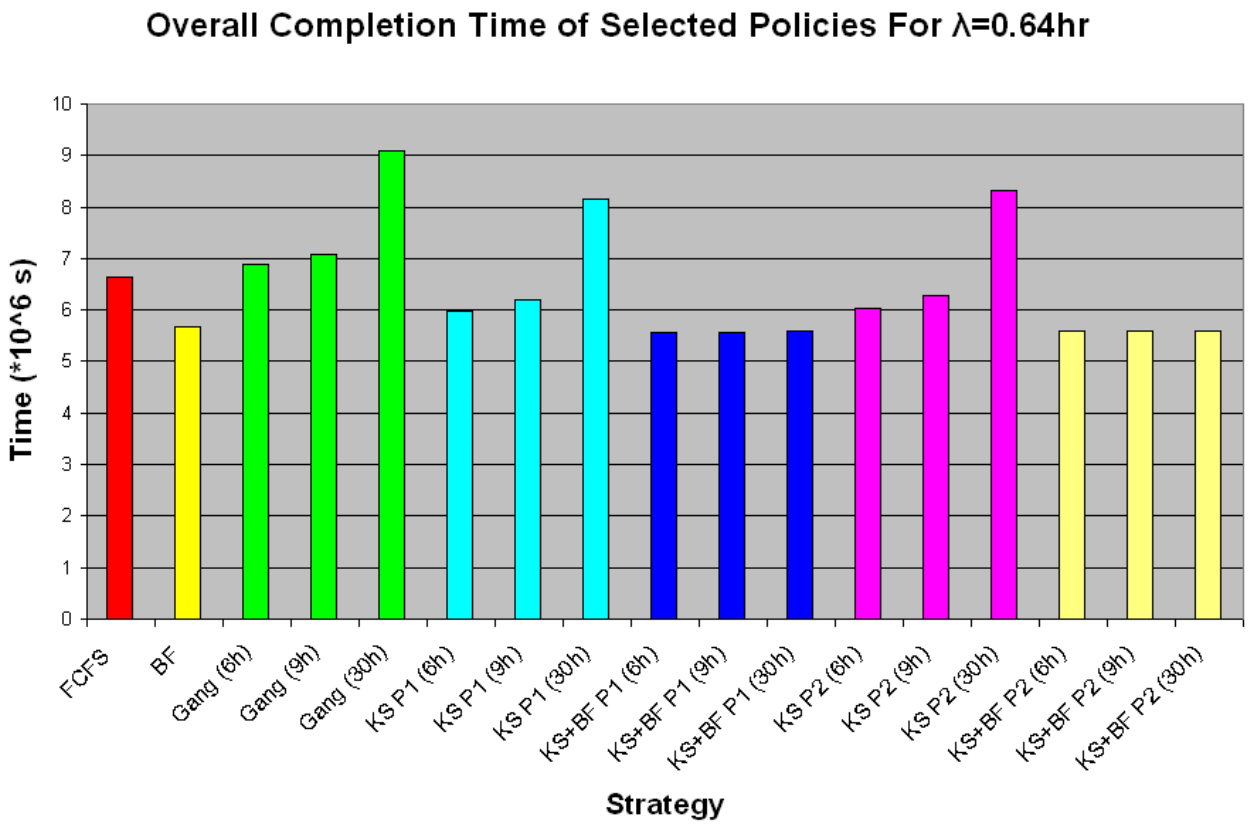
Correlation of Credit-Value-Metric with Speedup



University of Victoria

Final Task Completion Times

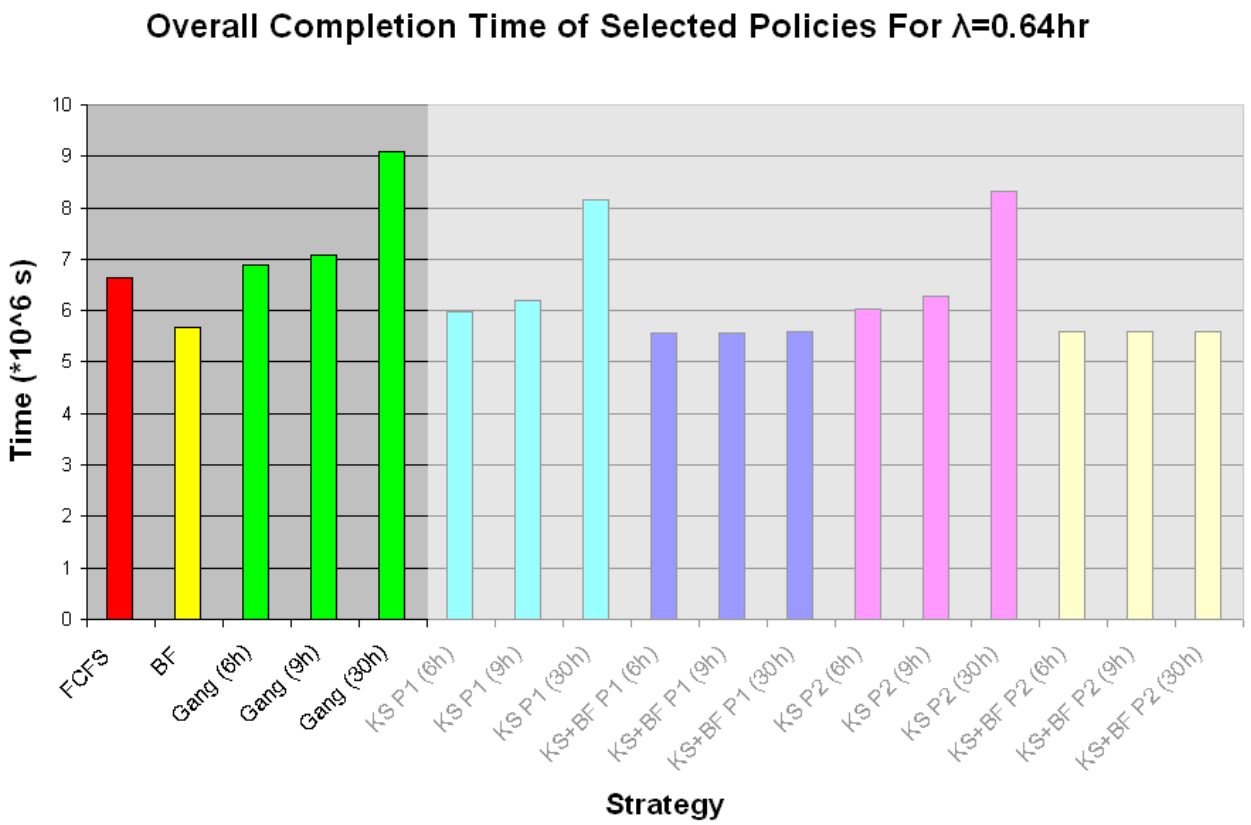
- Does the Knapsack increase the overall completion time?



University of Victoria

Final Task Completion Times

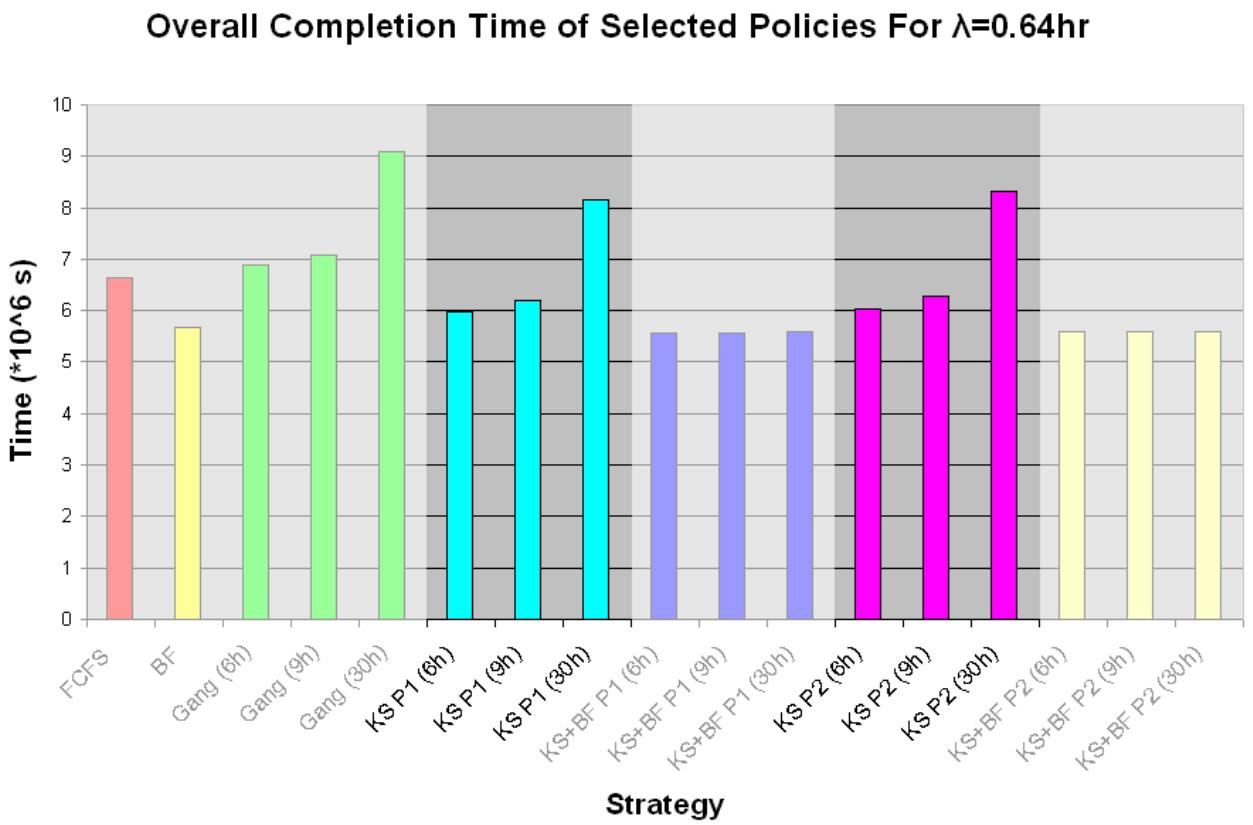
- Ref. Strategies: BF is best, gang struggles with large τ.



University of Victoria

Final Task Completion Times

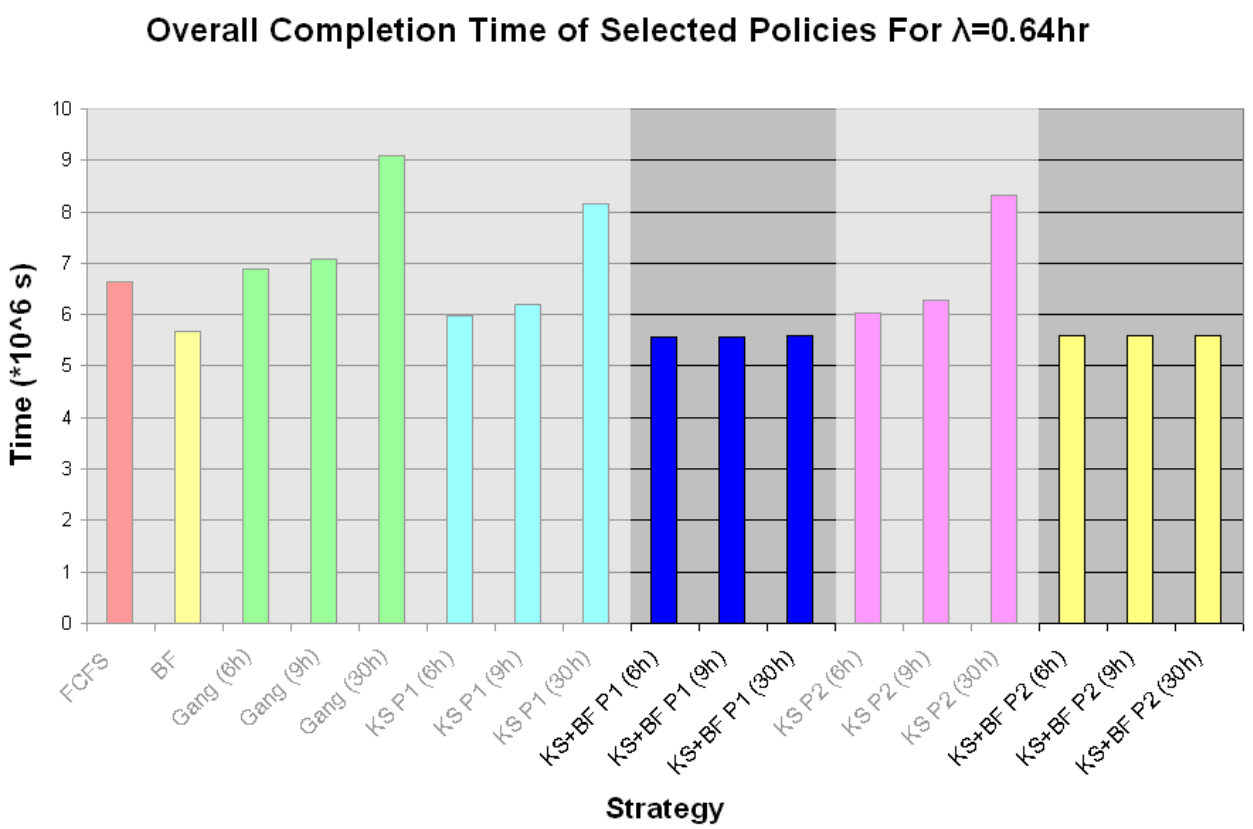
- Knapsack does better than Gang.



University of Victoria

KS+BF is Best and Provides QoS

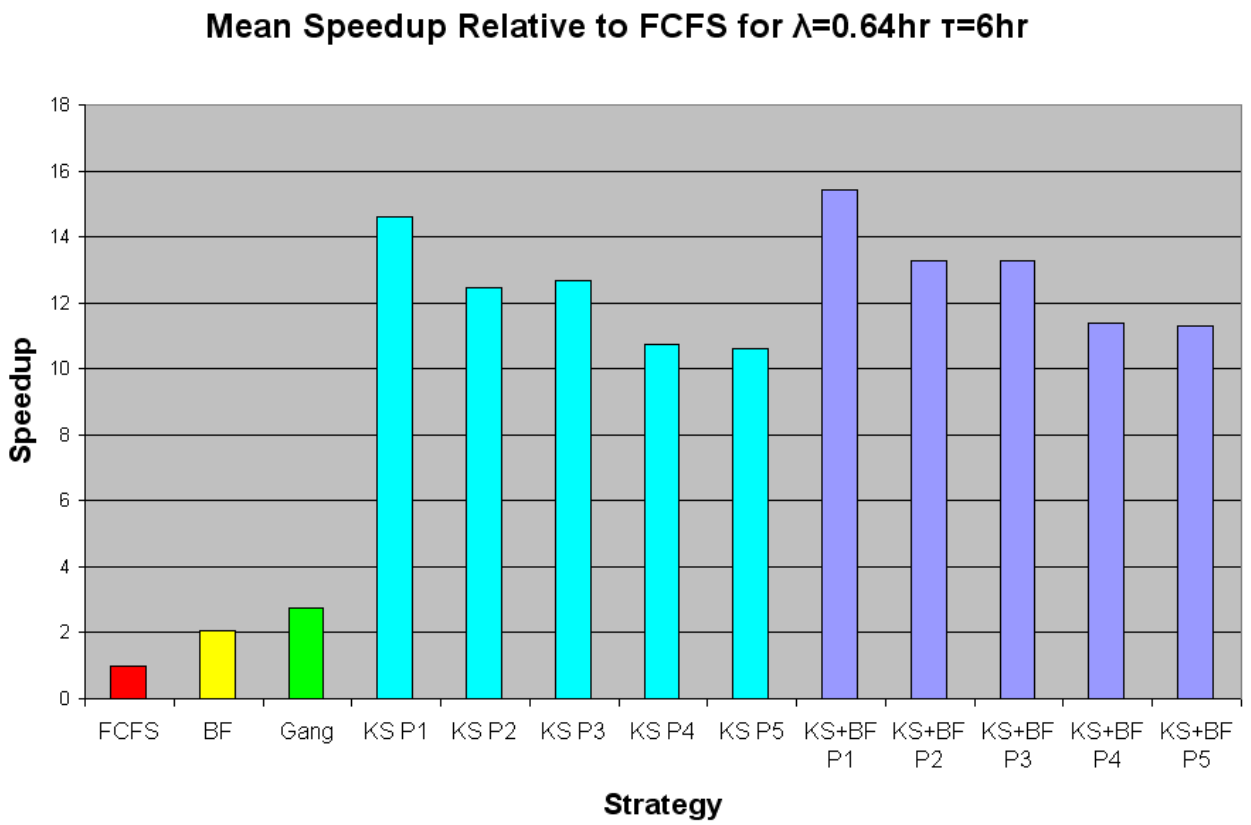
- KS+BF is best, even while providing QoS.



University of Victoria

Speedup Relative to FCFS

- Knapsack has the largest task speedup.



University of Victoria

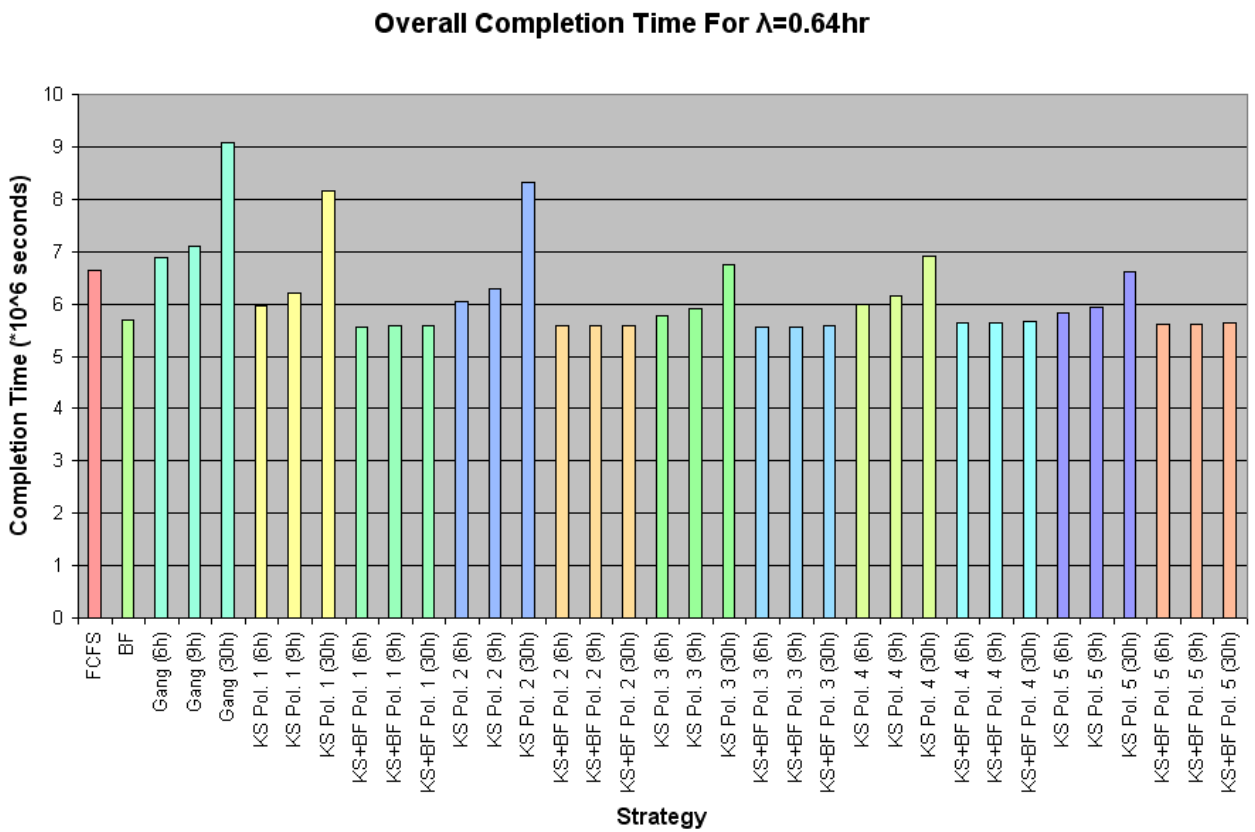
Conclusions

- Grid Resource Allocation problem can be formulated and solved as a knapsack problem.
- QoS-enabled policies can provide preferential allocation of high-value tasks while decreasing overall completion time.

Future Work

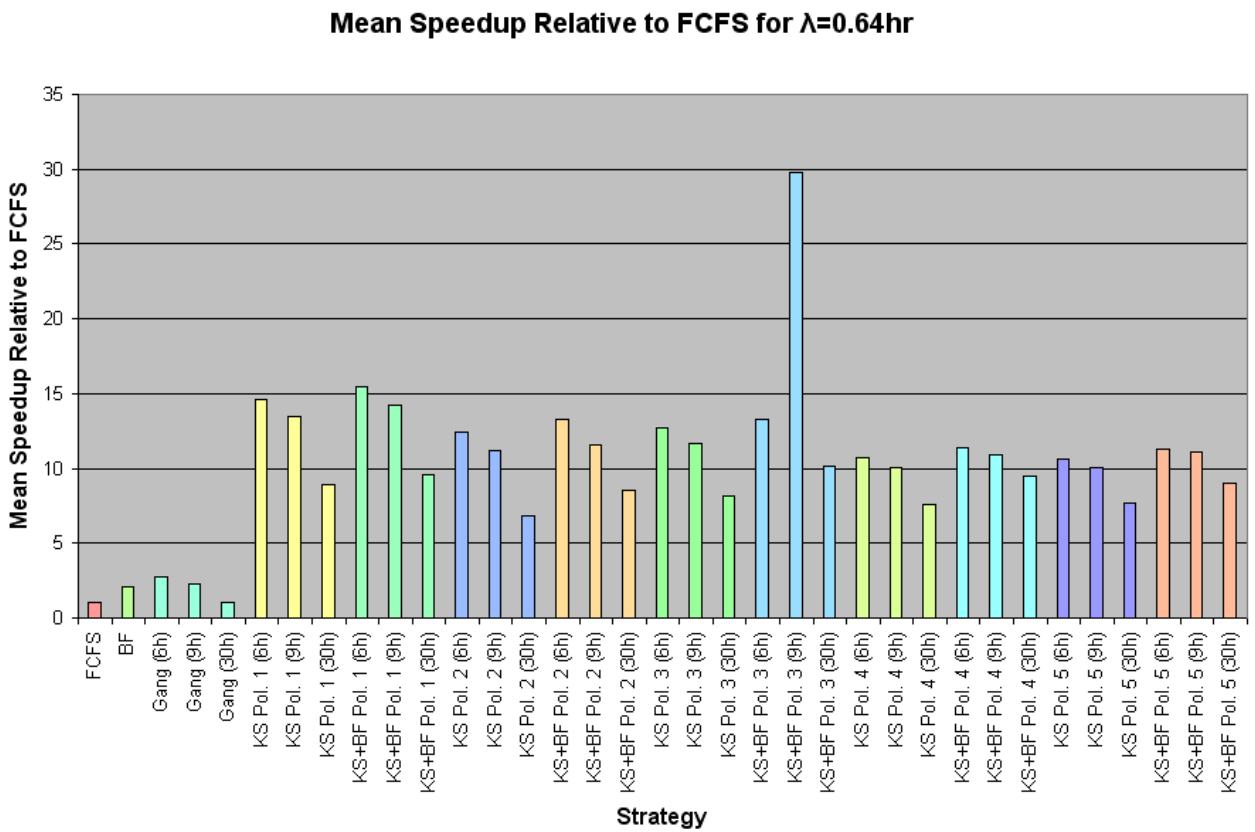
- Test the generality of the solution using NPACI workloads
- Evaluate the strategy with multiple resource types.

Final Task Completion Times



University of Victoria

Full Speedup Chart



University of Victoria