Service Level Agreement based Allocation of Cluster Resources: Handling Penalty to Enhance Utility

Chee Shin Yeo and Rajkumar Buyya

Grid Computing and **D**istributed **S**ystems (GRIDS) Lab.
Dept. of Computer Science and Software Engineering
The University of Melbourne, Australia
http://www.gridbus.org

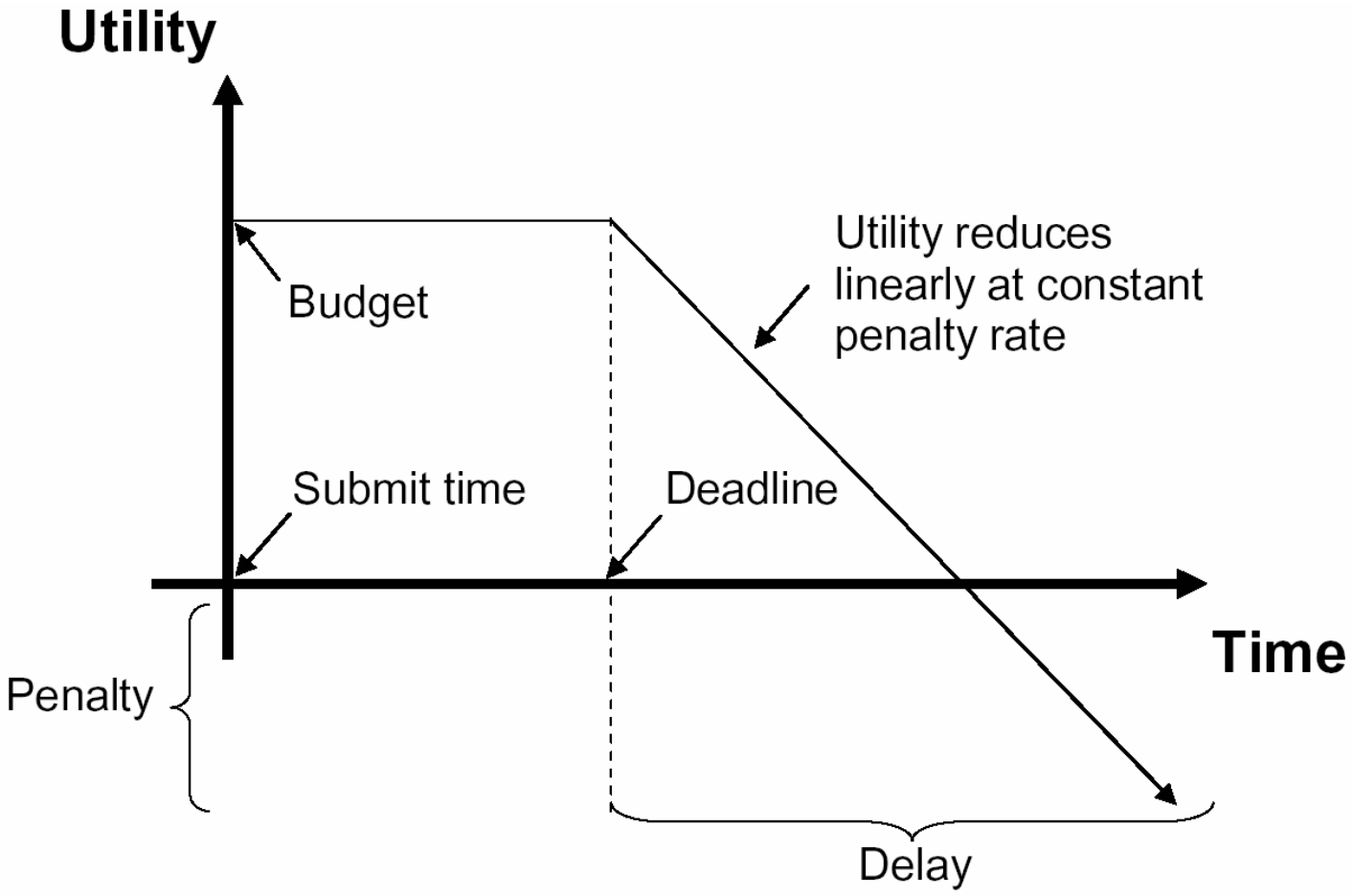
Problem

- Providing a service market via Service-oriented Grid computing
  - IBM's E-Business on Demand, HP's Adaptive Enterprise, Sun Microsystem's pay-as-you-go
  - Grid resources comprise clusters
- Utility-driven cluster computing
  - Service Level Agreement (SLA): differentiate different values and varying requirements of jobs depending on user-specific needs and expectations
  - Cluster Resource Management System (RMS) need to support and enforce SLAs

Proposal

- Current Cluster RMSs focus on overall job performance and system usage
- Using market-based approaches for utility-driven computing
- Utility based on users' willingness to pay
- Utility varies with users' SLAs
  - Deadline
  - Budget
  - Penalty

Impact of Penalty Function on Utility



Service Level Agreement (SLA)

- Delay
  - Delay = (finish_time – submit_time) - deadline
- Utility
  - Utility = budget – (delay * penalty_rate)
  - No Delay
    - Utility = Budget
  - Delay
    - 0 < Utility < Budget
    - Utility < 0
- LibraSLA – considers risk of penalties
  - Proportional share
  - Considers job properties
    - Run time
    - Number of processors

LibraSLA

- SLA based Proportional Share with Utility Consideration
  - Users express utility as budget or amount of real money
  - Focuses on resource allocation (not elaborating on other market concepts such as user bidding strategies or auction pricing mechanisms)
  - Users only gain utility and pay for service upon job completion (may be penalty)

LibraSLA

- Estimated run time provided during job submission is accurate
- Deadline of a job > its estimated run time
- SLA does not change after job acceptance
- Users submit jobs thru Cluster RMS only
- Cluster nodes may be homogeneous or heterogeneous
- Time-shared scheduling supported at nodes

LibraSLA

- Proportional Share of a job *i* on node *j*
  - Deadline and Run time
  - $share_{ij} = \frac{remain\_runtime_{ij}}{remain\_deadline_{ij}}$
- Total share for all jobs on a node *j*
  - $total\_share_j = \sum_{i=1}^{n_j} share_{ij}$
  - Delay when total_share > maximum processor time of node

LibraSLA

- Return of a job *i* on node *j*
  - $return_{ij} = utility_{ij}/runtime_i/deadline_i$
  - Return < 0 if utility < 0
  - Favors jobs with shorter deadlines
  - Higher penalty for jobs with shorter deadlines
- Return of a node *j*
  - $return_j = \sum_{i=1}^{n_j} return_{ij}$
  - Lower return indicates overloading

LibraSLA

- Admission Control (Accept new job or not?)
  - Determines return of each node if new job is accepted
  - Node is suitable if
    - It has higher return
    - It can satisfy HARD deadline if required
  - New job accepted if enough suitable nodes as requested
  - Accepted new job allocated to nodes with highest return

LibraSLA

- Determines return of a node
  - Determines total share of processor time to fulfill deadlines of all its allocated jobs and new job
  - Identifies job with highest return
  - Gives additional remaining share to job with highest return (if any)
  - If insufficient processor time, only job with highest return and jobs with hard deadlines are not delayed;

    jobs with soft deadlines are delayed proportionally
  - Returns of these delays computed accordingly

Performance Evaluation: Simulation

- Simulated scheduling for a cluster computing environment using the GridSim toolkit

  (http://www.gridbus.org/gridsim)

Experimental Methodology: Trace Properties

- Feitelson's Parallel Workload Archive
  (http://www.cs.huji.ac.il/labs/parallel/workload)
- Last 1000 jobs in SDSC SP2 trace
  - Average inter arrival time:
    2276 secs (37.93 mins)
  - Average run time:
    10610 secs (2.94 hrs)
  - Average number of requsted processors:
    18

Experimental Methodology: Cluster Properties

- SDSC SP2:
  - Number of computation nodes:
    128
  - SPEC rating of each node:
    168
  - Processor type on each computation node:
    RISC System/6000
  - Operating System:
    AIX

Experimental Methodology: SLA Properties

- 20% - HIGH urgency jobs
  - HARD deadline type
  - LOW deadline/runtime
  - HIGH budget/f(runtime)
  - HIGH penalty_rate/g(runtime)

  where f(runtime) and g(runtime) are functions representing the MINIMUM budget and penalty rate for the user-specified runtime

Experimental Methodology: SLA Properties

- 80% - LOW urgency jobs
  - SOFT deadline type
  - HIGH deadline/runtime
  - LOW budget/f(runtime)
  - LOW penalty_rate/g(runtime)

  where f(runtime) and g(runtime) are functions representing the MINIMUM budget and penalty rate for the user-specified runtime

Experimental Methodology: SLA Properties

- High:Low ratio
  - Eg. Deadline high:low ratio is the ratio of means for high deadline/runtime (low urgency) and low deadline/runtime (high urgency)
  - Deadline high:low ratio of 7
  - Budget high:low ratio of 7
  - Penalty Rate high:low ratio of 4

Experimental Methodology: SLA Properties

- Values normally distributed within each HIGH and LOW
  - deadline/runtime
  - budget/f(runtime)
  - penalty_rate/g(runtime)
- HIGH and LOW urgency jobs randomly distributed in arrival sequence

Experimental Methodology: Performance Evaluation

- Arrival delay factor
  - Models cluster workload thru inter arrival time of jobs
  - Eg. arrival delay factor of 0.01 means a job with 400 s of inter arrival time now has 4 s
- Mean factor
  - Denotes mean value for normal distribution of deadline, budget and penalty rate SLA parameters
  - Eg. Mean factor of 2 means having mean value double that of 1 (ie. higher)

Experimental Methodology: Performance Evaluation

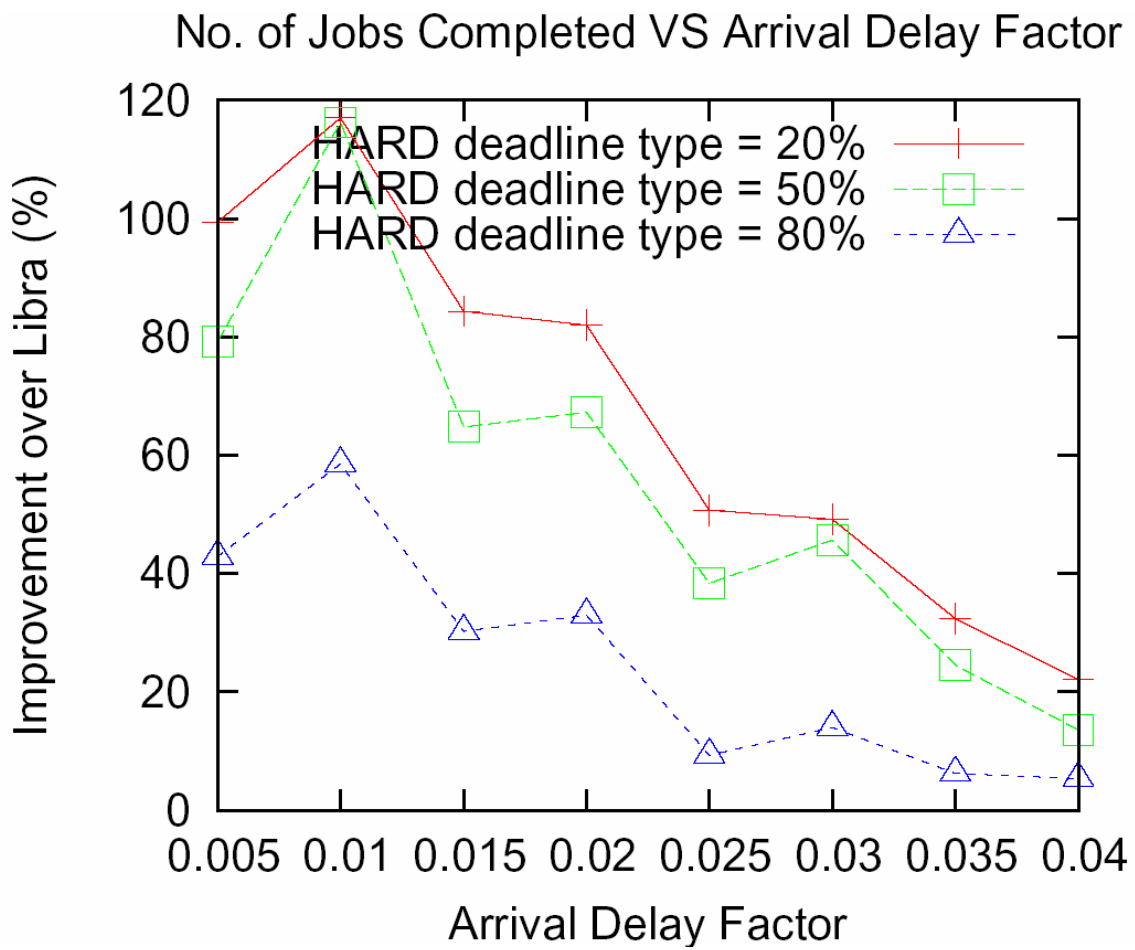
- Comparison with Libra
  - Assumes HARD deadline
  - Selects nodes based on BEST FIT strategy

    (ie. nodes with least available processor time after accepting the new job are selected first)
- Evaluation Metrics
  - Number of jobs completed with SLA fulfilled
  - Aggregate utility achieved for jobs completed

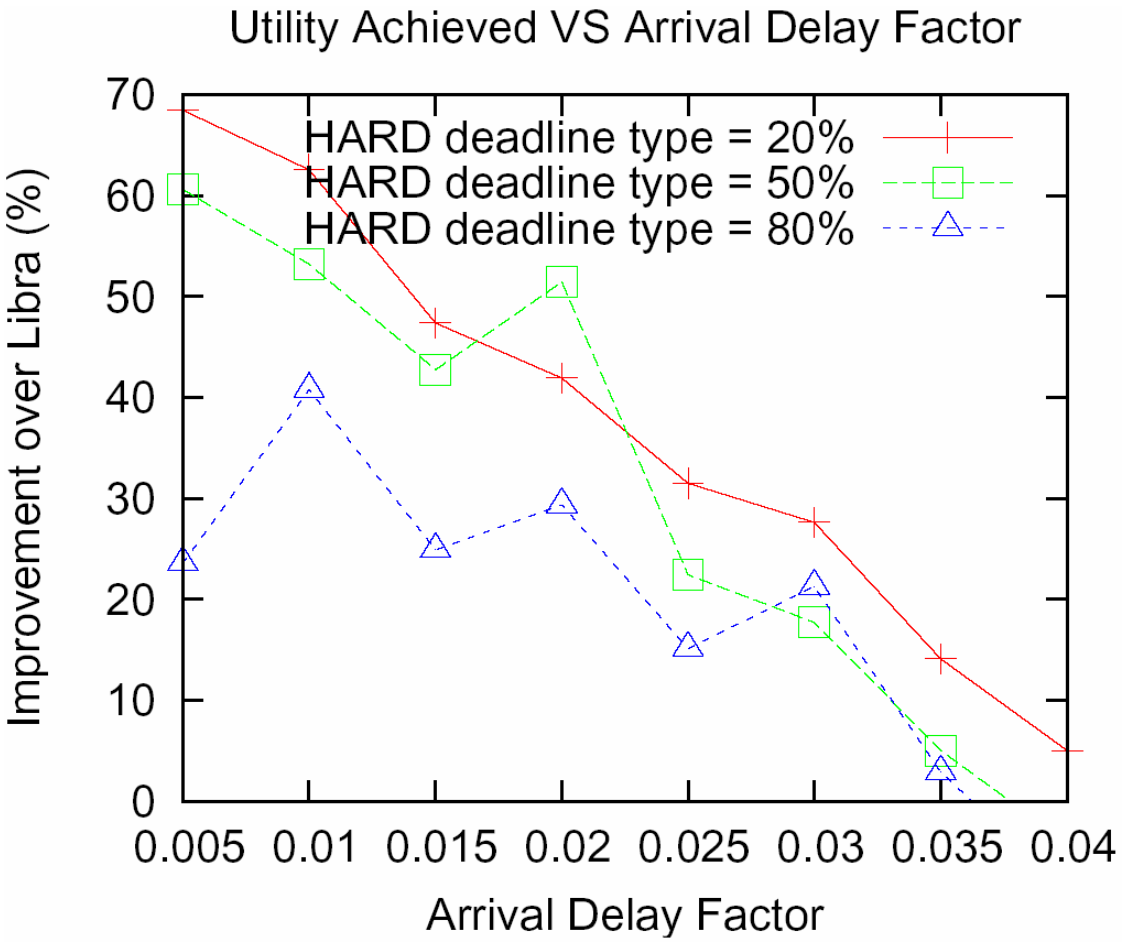
Performance Evaluation: Impact of Various SLA Properties

- Deadline type
  - Hard: no delay
  - Soft: can accommodate delay (Penalty rate determines limits of delay)
- Deadline
  - Time period to finish the job
- Budget
  - Maximum amount of currency user willing to pay
- Penalty rate
  - Compensate user for failure to meet deadline
  - Reflects flexibility with delayed deadline (higher penalty rate limits delay to be shorter)

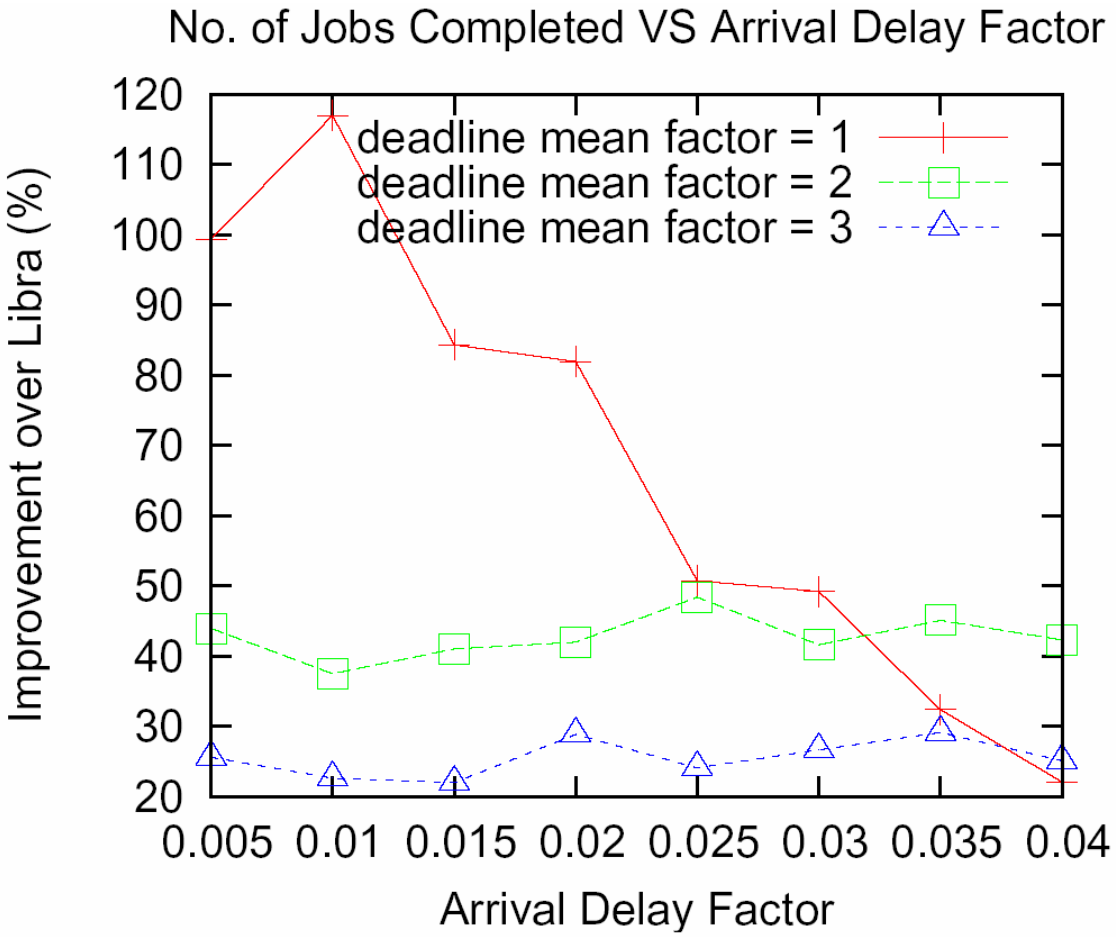
Deadline Type



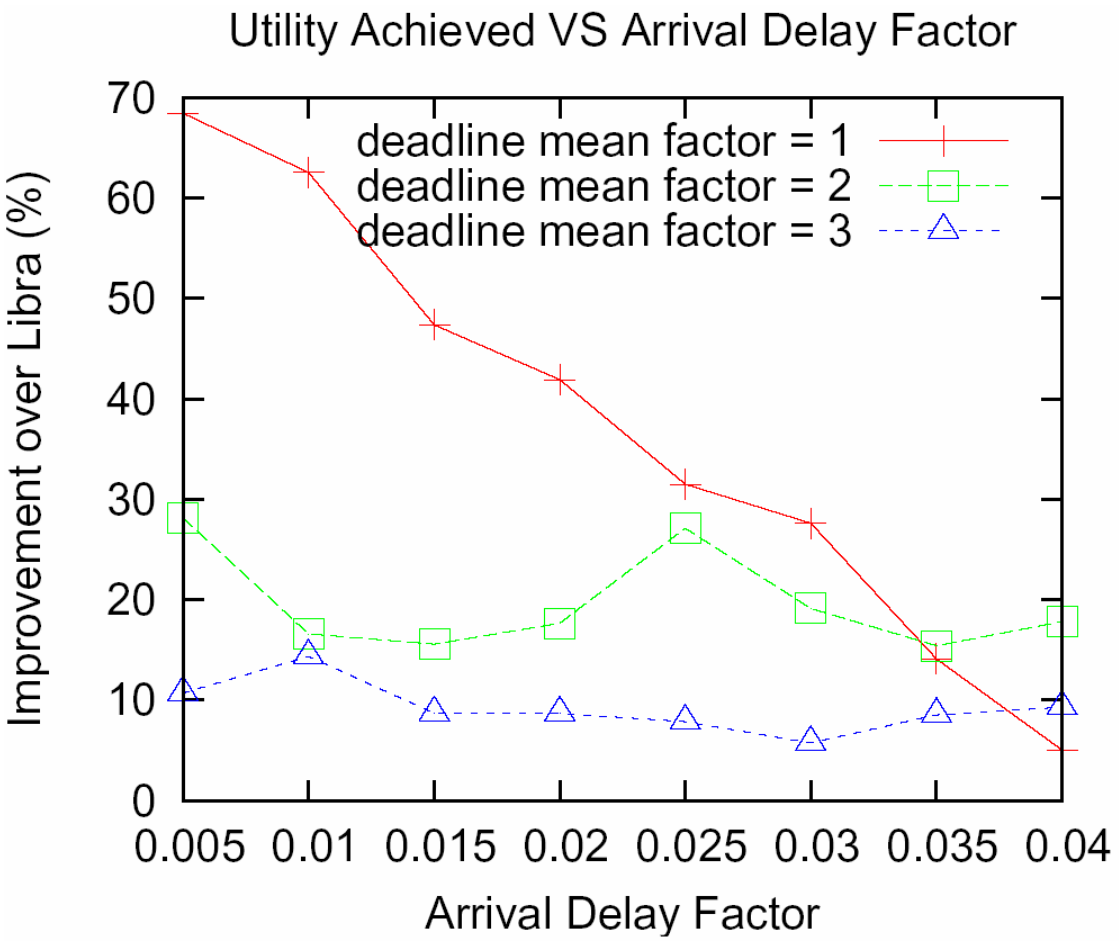
Deadline Type



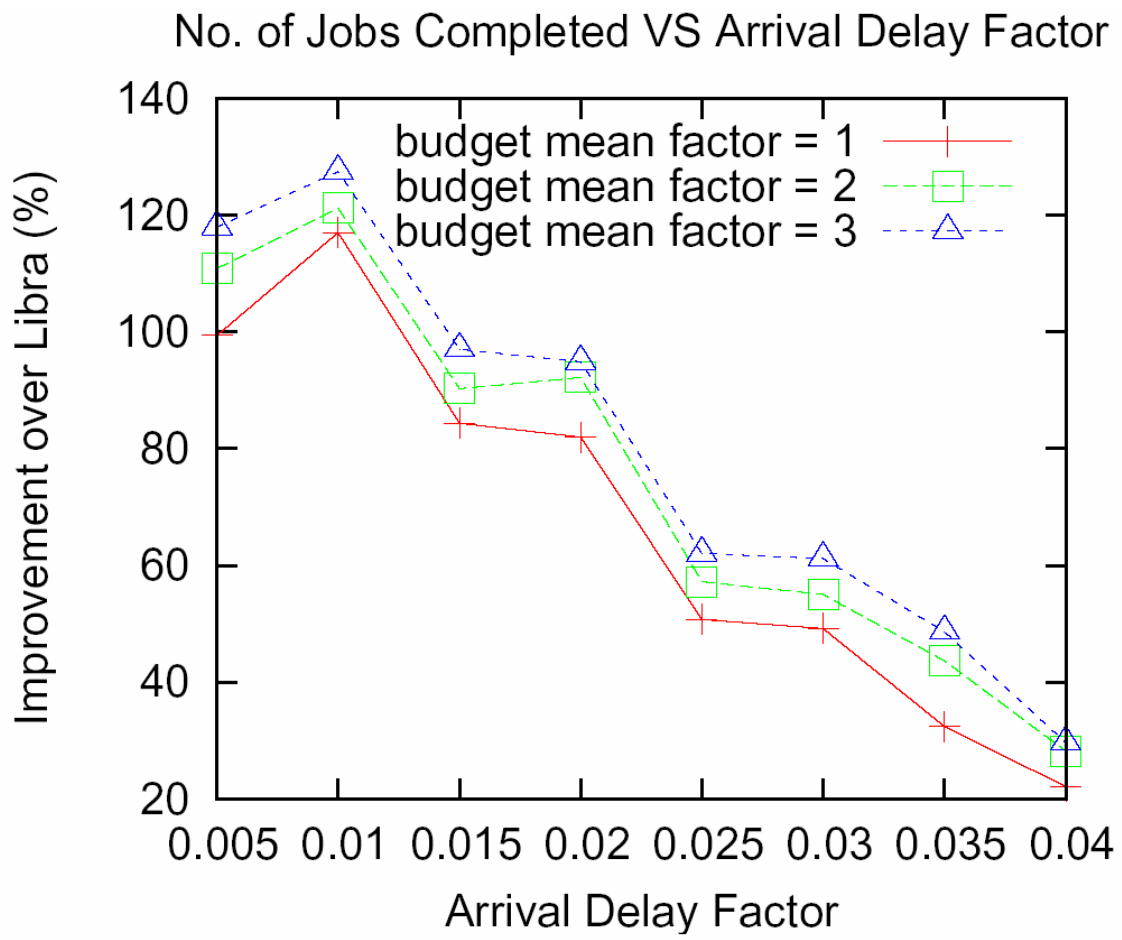
Deadline Mean Factor



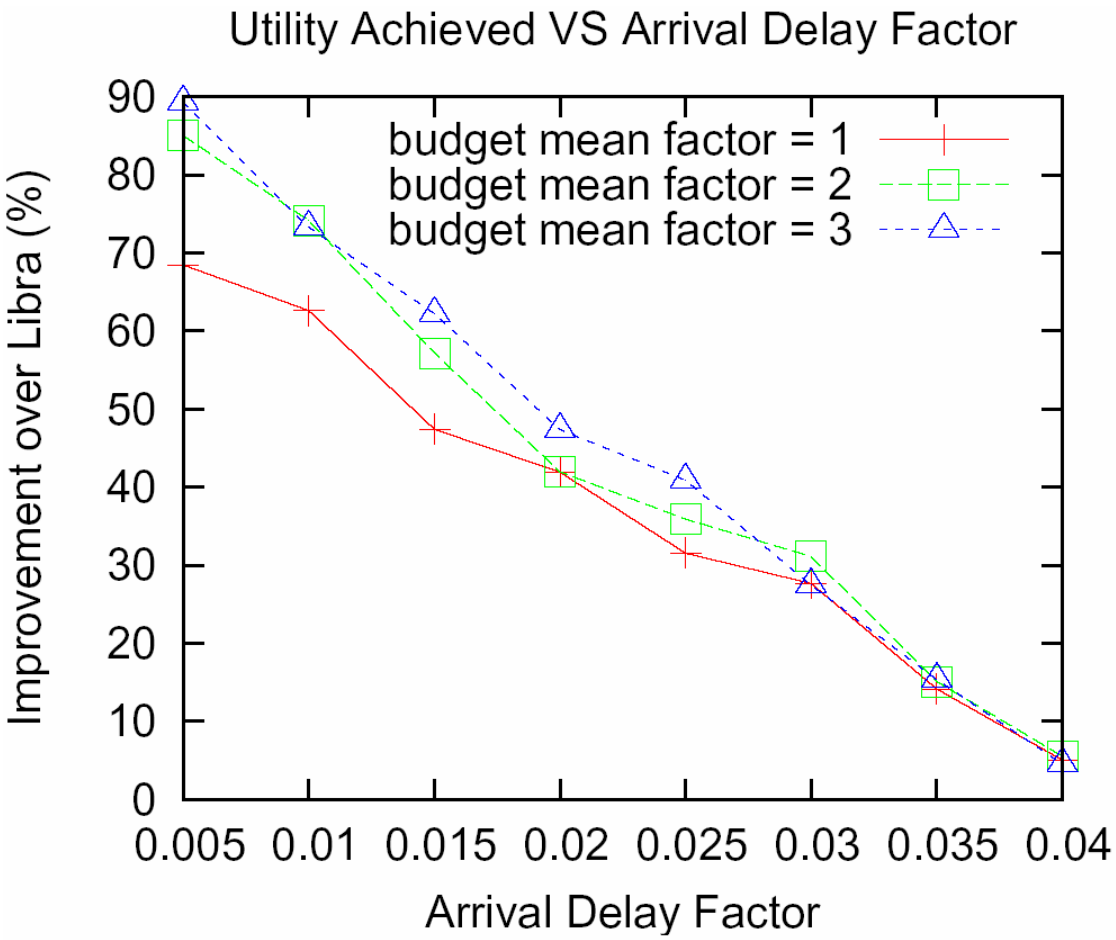
Deadline Mean Factor



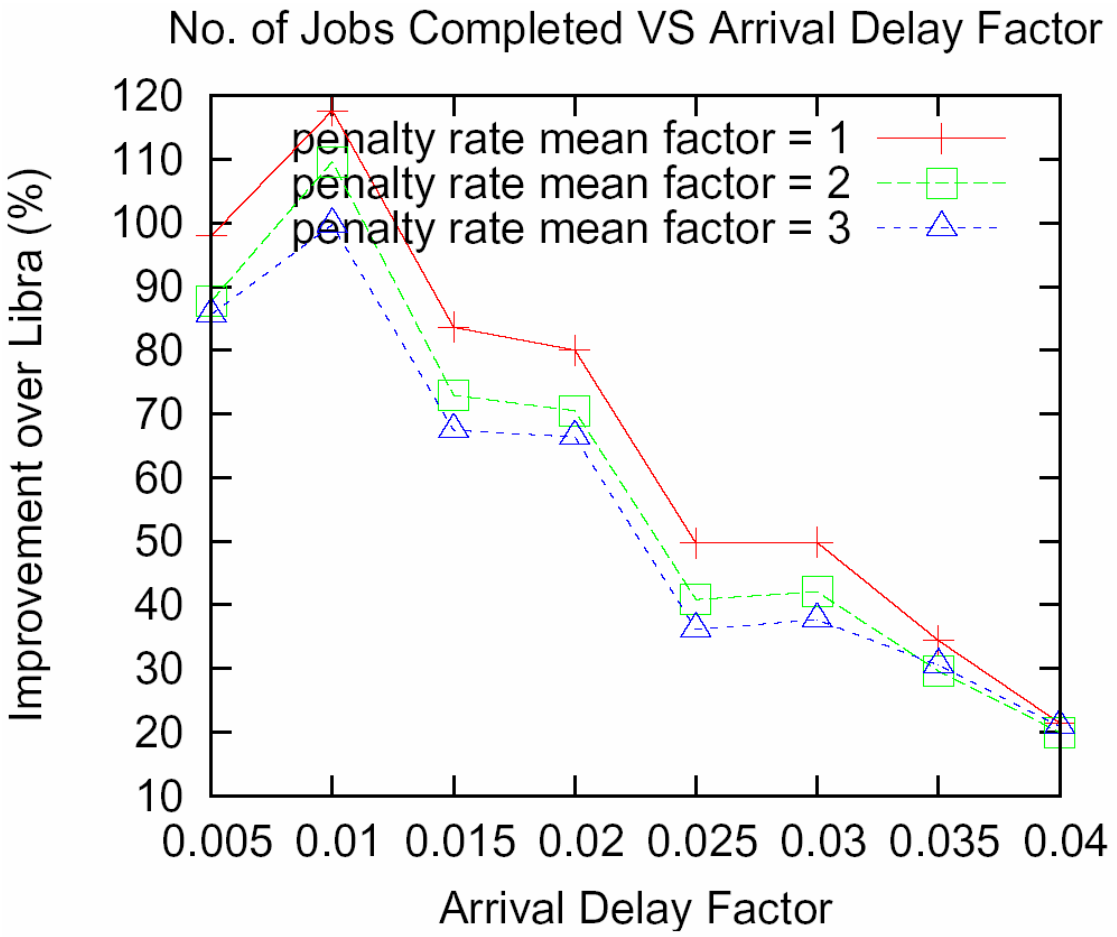
Budget Mean Factor



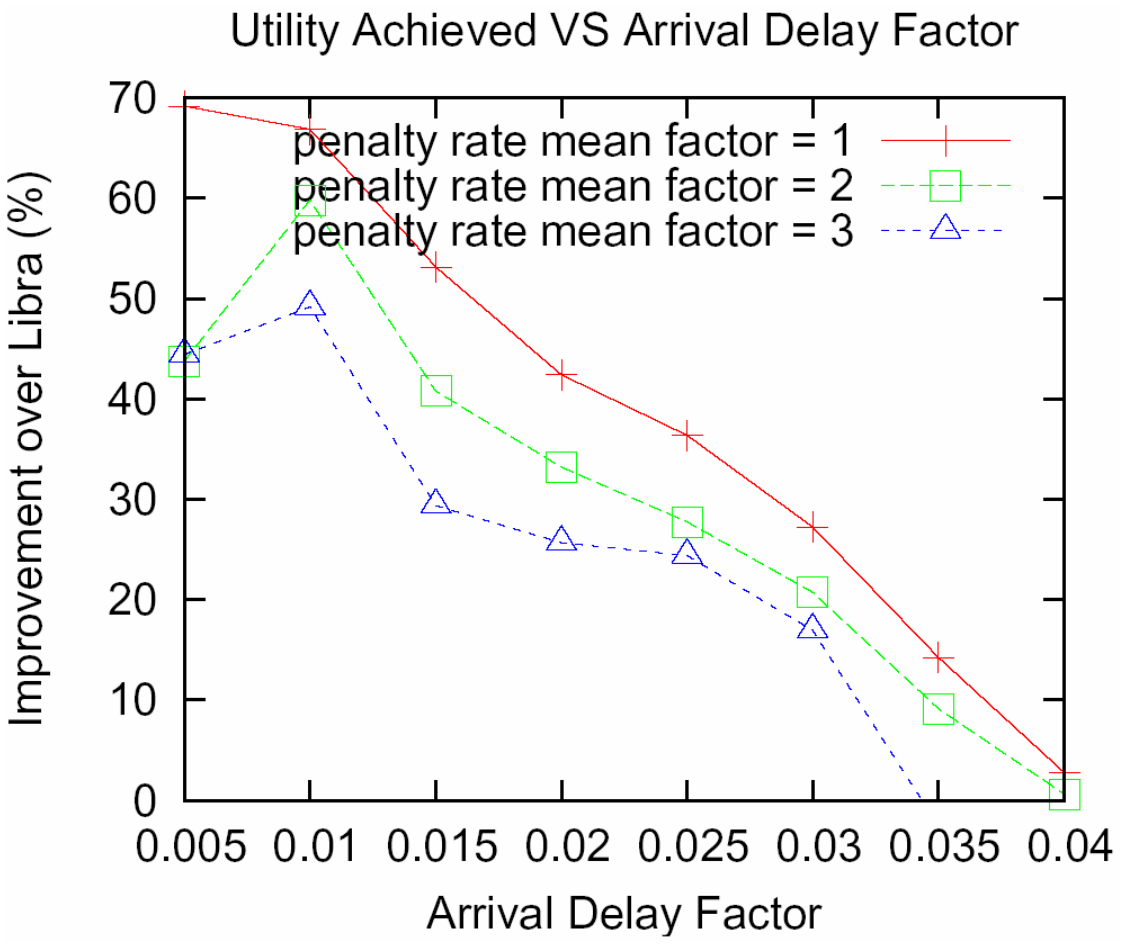
Budget Mean Factor



Penalty Rate Mean Factor



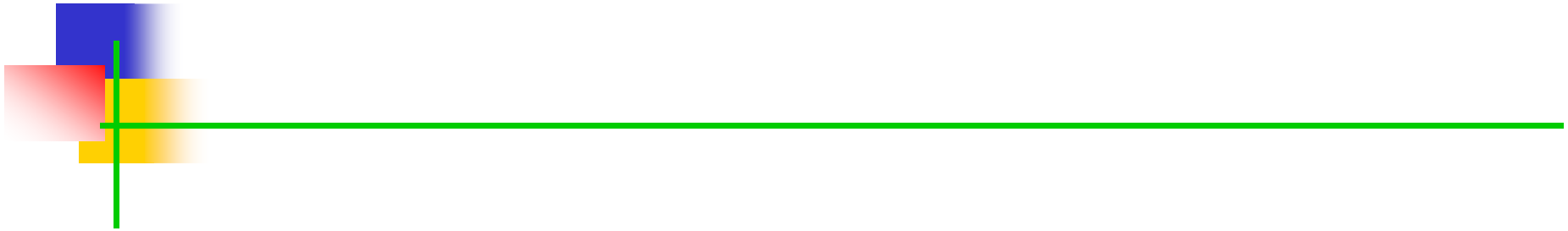
Penalty Rate Mean Factor



Conclusion

- Importance of handling penalty in SLAs
- LibraSLA
  - Fulfill more SLAs thru soft deadlines
  - Minimizes penalties to improve utility
- SLA with 4 parameters
  - (i) Deadline Type (ii) Deadline (iii) Budget (iv) Penalty Rate
- Need to support
  - Utility-driven cluster computing
  - Service-oriented Grid computing

End of Presentation



Questions ?