

Batch Job Scheduling with Realistic Utility Functions

(Paper: Precise and Realistic Utility Functions for User-Centric Performance Analysis of Schedulers)

Cynthia B. Lee
Allan E. Snavely

Department of Computer Science and Engineering
and San Diego Supercomputer Center

University of California, San Diego



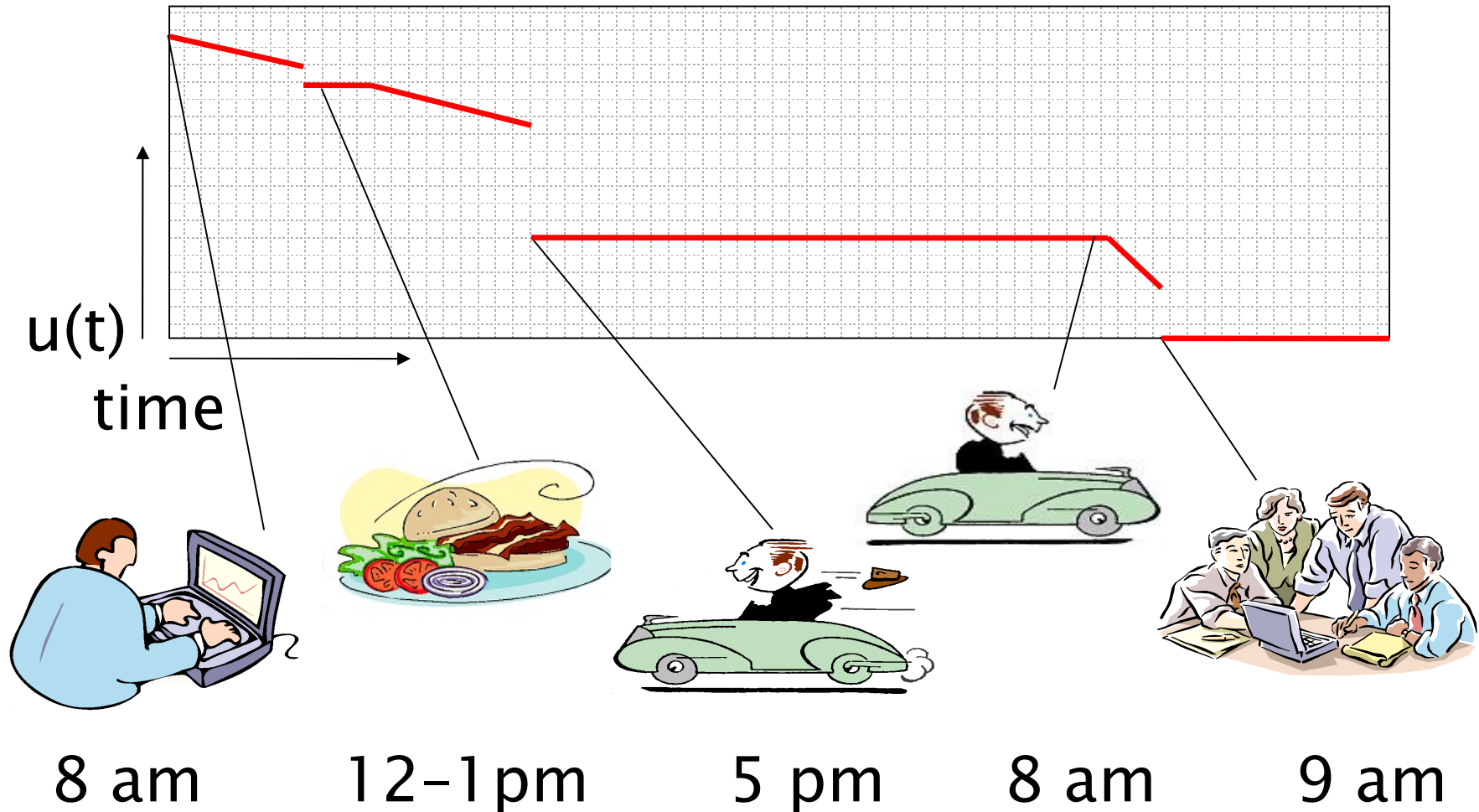
Introduction

- Economics-based resource management of HPC and grid systems is an area of active research
- Two main weaknesses in current research:
 - 1) Not using real workloads including 'messiness' like requested runtime different from actual, etc.
 - 2) Model of users' valuation of jobs and scheduler service not based on any actual data or user input

Contributions of This Work

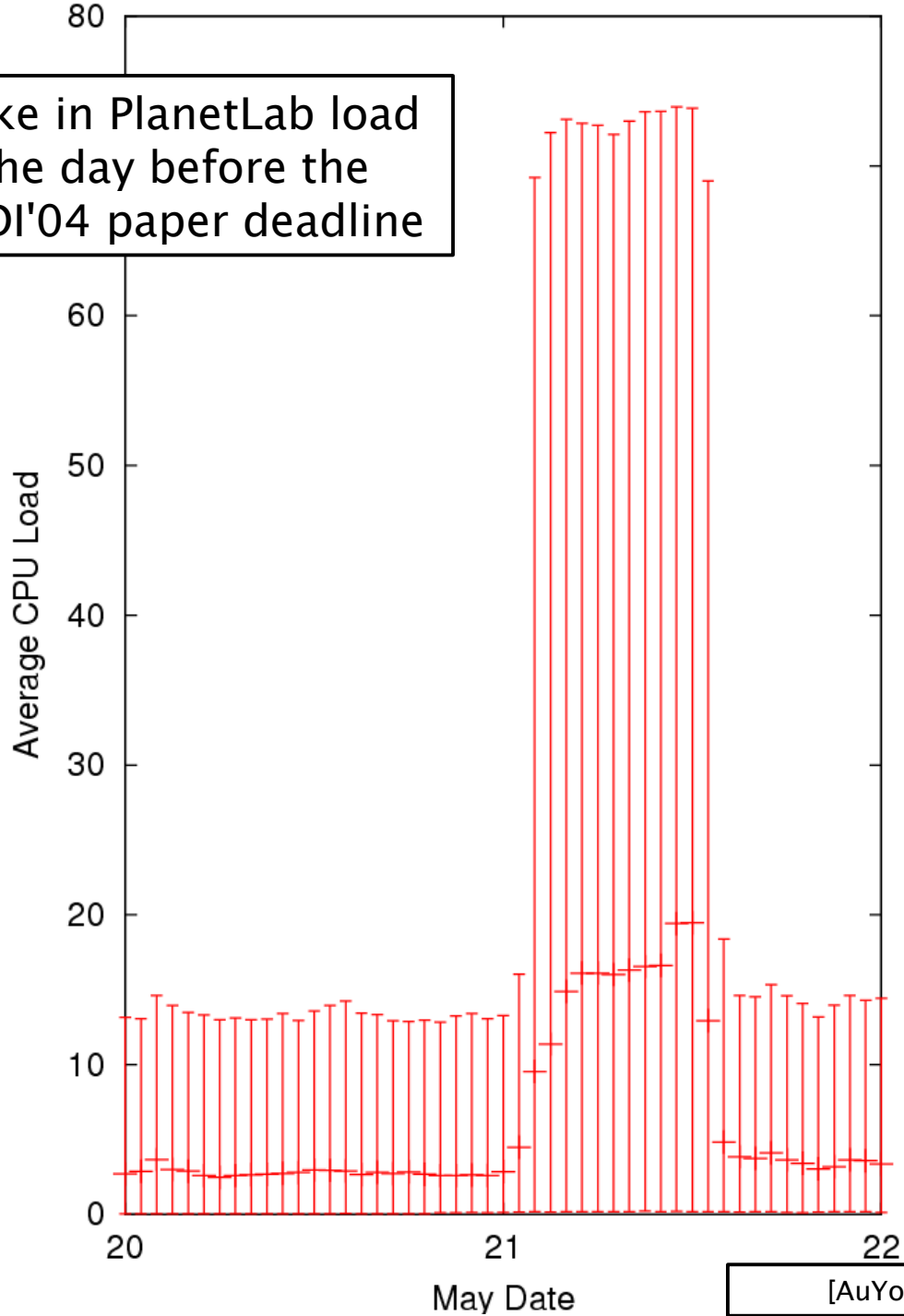
- 1) Proposed augmenting the *Standard Workload Format* for supercomputer logs [Dror Feitelson et al.] to include arbitrarily-shaped utility functions (could be user-provided or synthetically generated)
- 2) Proposed a model for synthetically generating utility functions that draws on patterns from actual users' valuation of their jobs
- 3) Demonstrate an effective prototype scheduler that uses aggregate utility as an explicit objective function

What is a Utility Function?



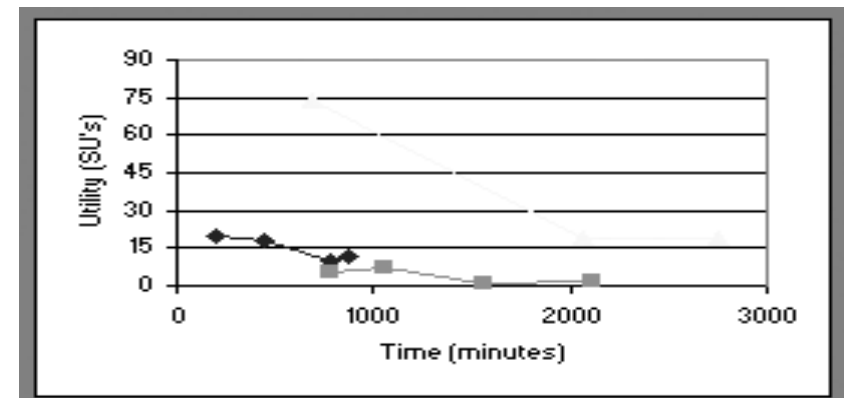
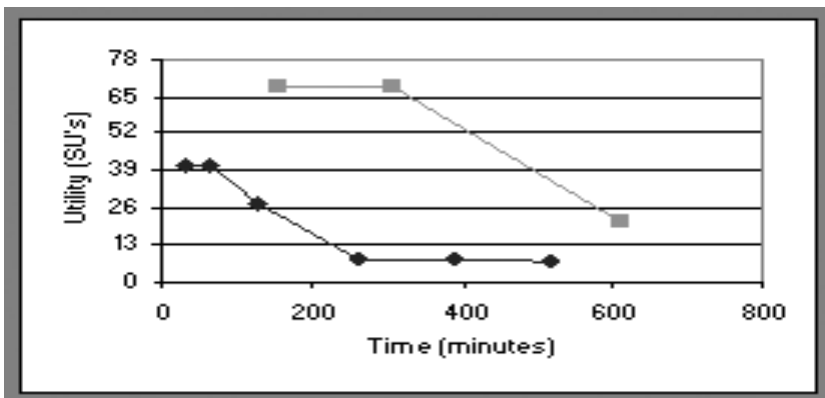
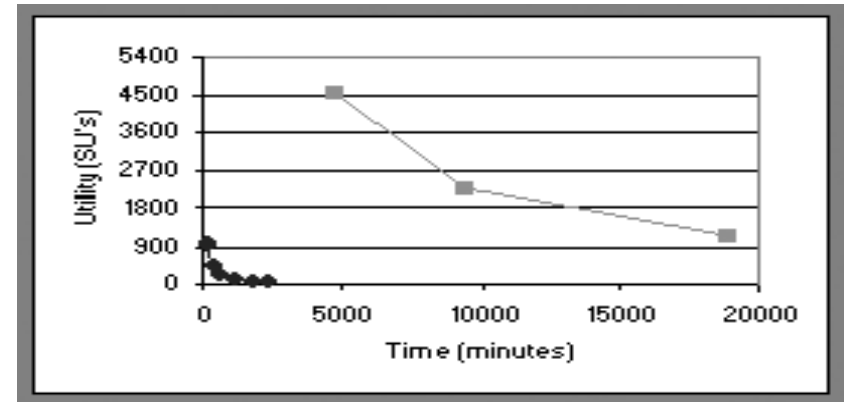
Other factors: coordinate with other grid sites or sensors, **paper deadlines**, weather and hurricane prediction, ...

Spike in PlanetLab load
the day before the
OSDI'04 paper deadline

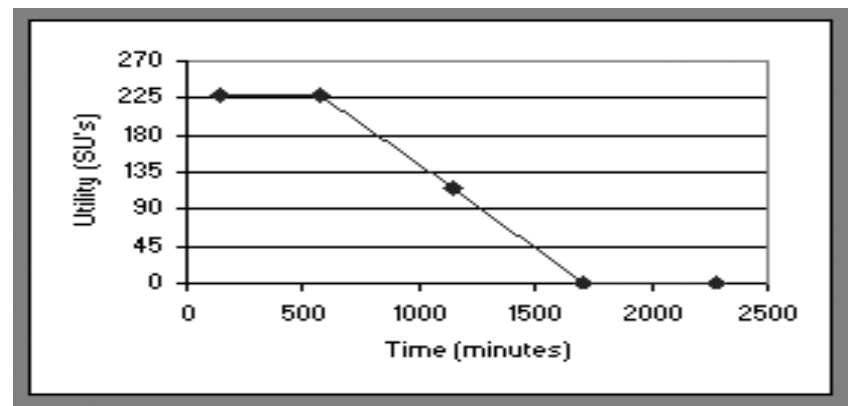
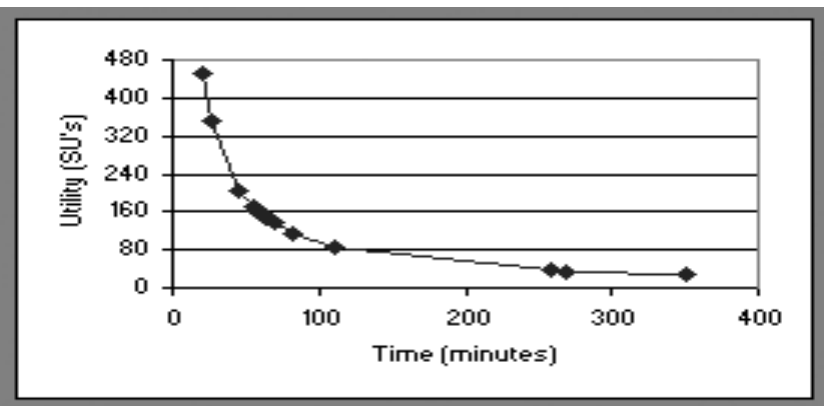
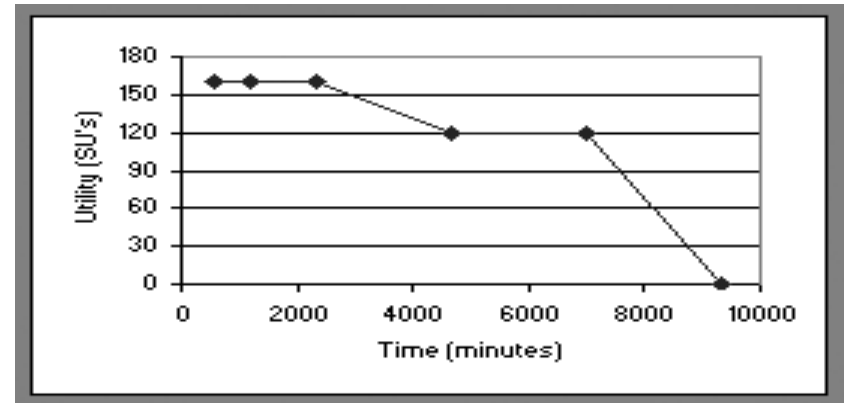
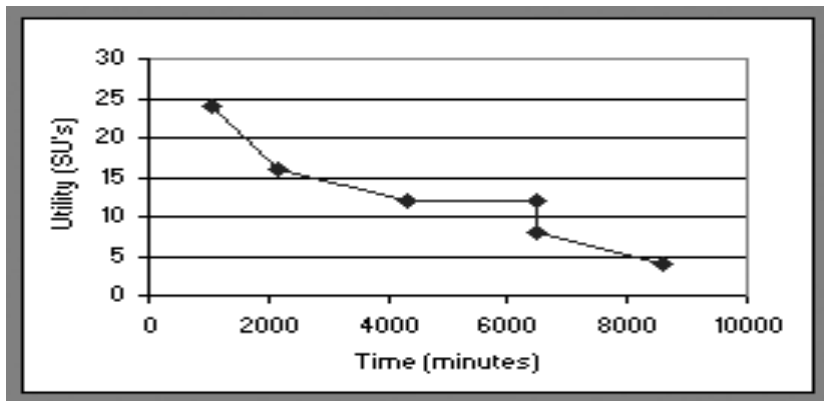


Real Users' Functions

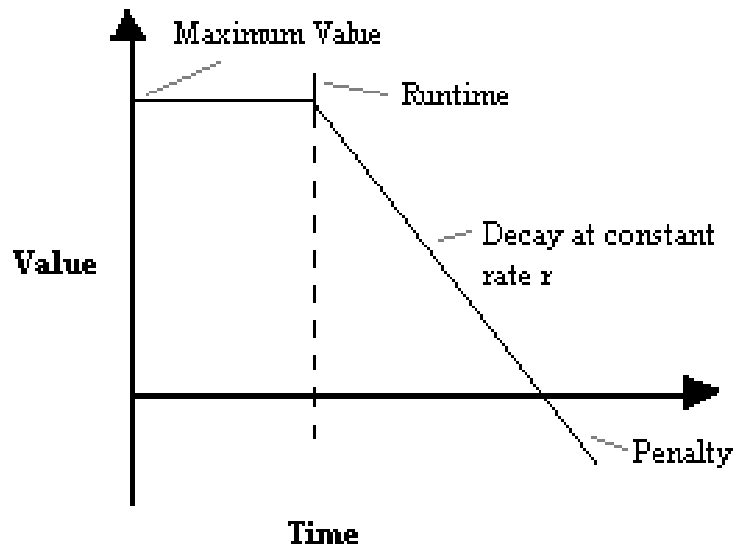
- Randomly-selected users of SDSC systems provided these data points for jobs they were submitting
- Utility is in terms of the SDSC charge unit ("SU")



More Real Users' Functions



Existing Model

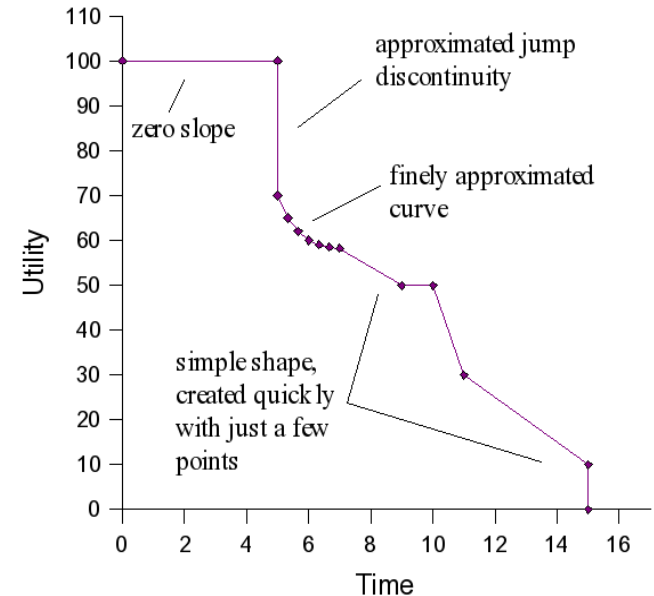


[Used by e.g. Chun and Culler 2002, and Irwin, Grit, Chase 2004]

Proposed Model

To use Aggregate Utility, utility functions needed *for all jobs*

- ✓ Propose to store function as series of *(time, value)* pairs appending each line of *Standard Workload Format*, allowing arbitrarily-shaped functions

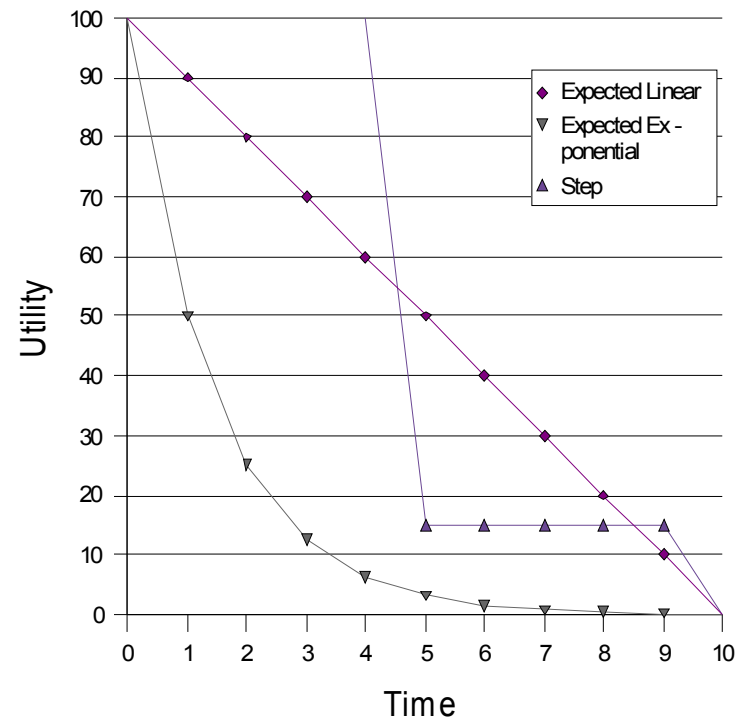


Job ID	Submit Time	Req. Time	Run Time	Nodes	...	Utility		
1						Time	Value	...
2								
3								
...								

Absent real data collected from users for each job, we need a model for synthetic generation...

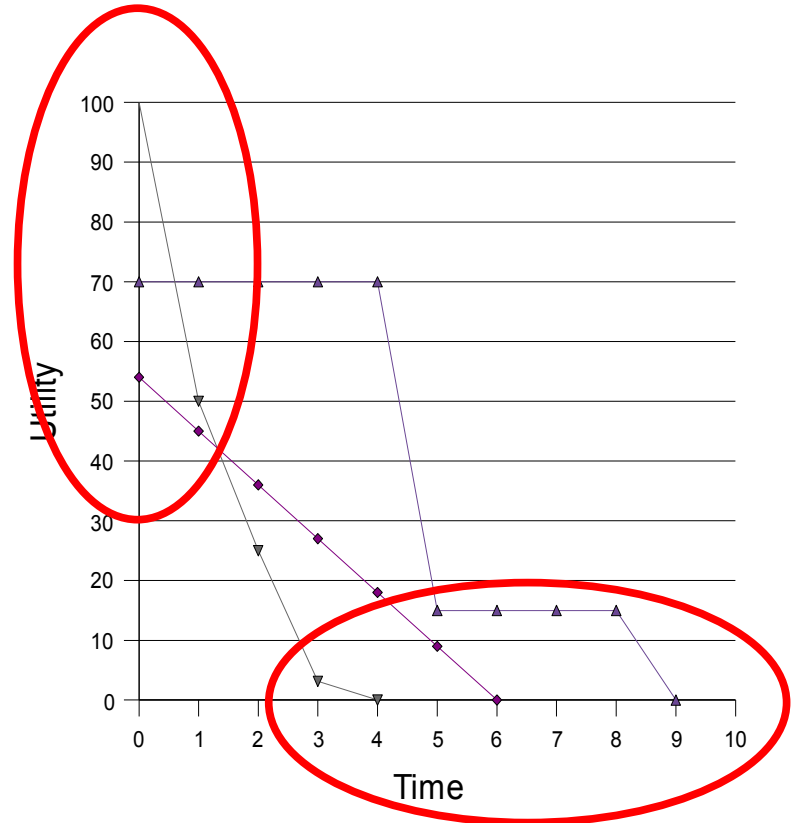
Modeling Three Distinct Decay Patterns

- *Expected Linear*
- *Expected Exponential*
- *Step*
- ✓ “Expected” refers to the fact that each point is chosen randomly (i.e. Most won't follow the pattern as cleanly as shown here)



Start Values and Deadlines

- User-provided priority (queue) from the log controls the starting (maximum) job value
- Distribution of actual wait times from the log controls the deadline (when the value goes to zero)



Metric: Aggregate Utility

$$\textit{Aggregate utility} = \sum_{j \in \textit{Jobs}} \textit{utility}_j(\textit{turnaround_time}_j)$$

- Reflects administrator's priorities
 - allocation of funds (“SUs”/Monopoly money) to users at the beginning of the fiscal [year/quarter/month/etc]
- Reflects users' personal input
 - how they choose to spend their funds
- Enables more comprehensive evaluation and comparison of *all* job scheduling algorithms

Parallel Job Scheduling

Explicitly by Utility Function

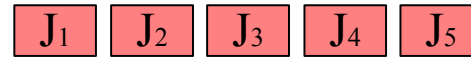


Finding the best solution is NP-hard

- “Tennis Court Scheduling” (human-powered)
 - *Still practiced occasionally at most centers (officially and not) -- a phone call to sys admins gets a job a reservation or to the front of the queue*
- Custom Heuristics
 - *Sort by current value, or a combination of start value and slope [Chun and Culler 2002; Irwin, Grit, Chase 2004]*

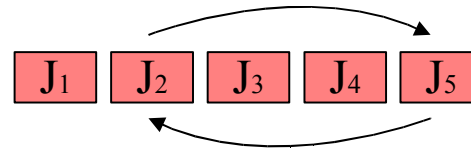
Genetic Algorithm Scheduler

- Individuals:



- permutations of the job queue ordering

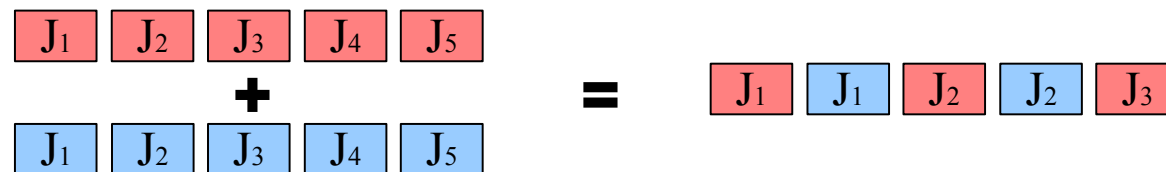
- Mutation:



- swap two randomly-selected jobs

- Reproduction:

- zipper-like merging of parents (skip duplicates)



- Fitness: global utility of resulting schedule (approx.)

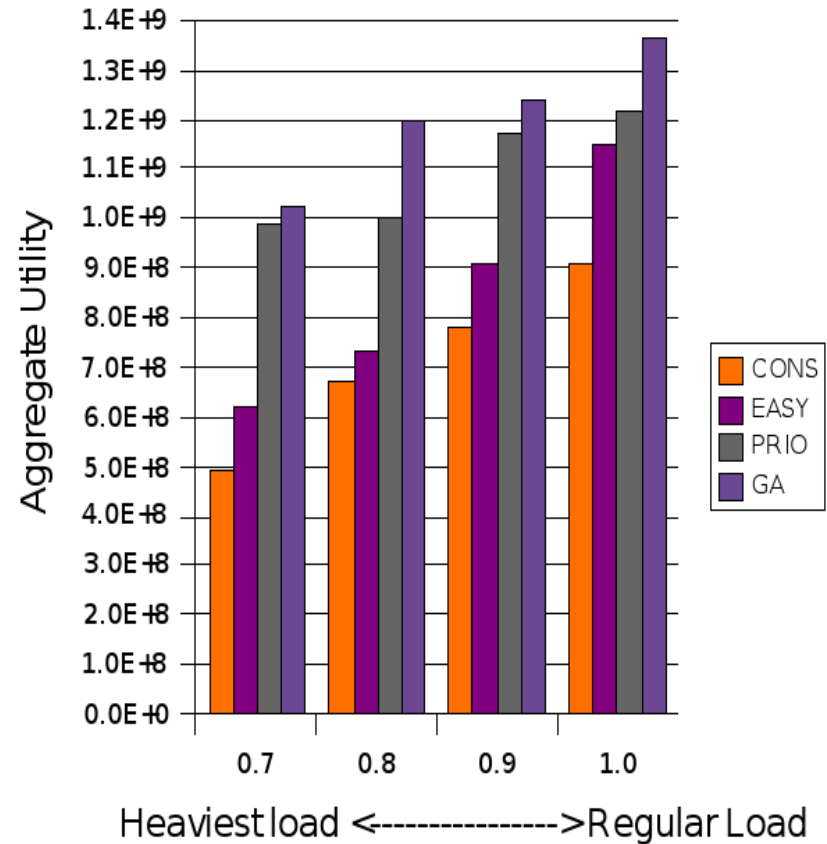
Results

Schedulers compared:

- CONS = Conservative Backfilling
- EASY = Aggressive Backfilling
- PRIO = Priority FIFO (typical supercomputer priority scheduler)
- GA = genetic algorithm

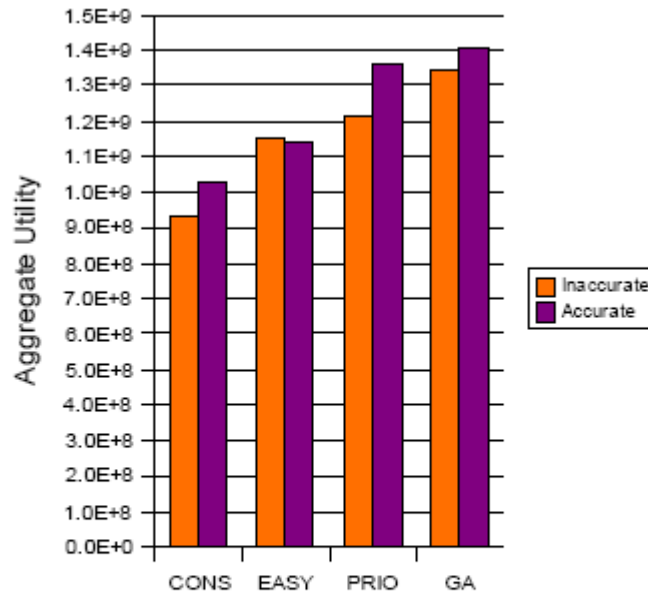
Workload is SDSC-BLUE from the Parallel Workloads Archive (Dror Feitelson)

Load modified by scaling inter-arrival times

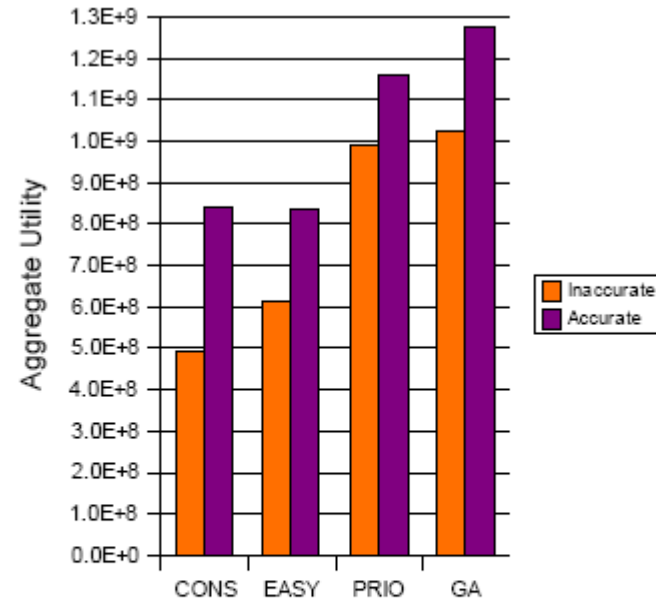


Accurate and Inaccurate Runtimes

Normal Load



Heavy Load



Many, many more results in the paper...

Contributions of This Work

- 1) Proposed augmenting the *Standard Workload Format* for supercomputer logs to include arbitrarily-shaped utility functions
- 2) Proposed a model for synthetically generating utility functions that draws on patterns from actual users' valuation of their jobs
- 3) Demonstrated a genetic algorithm-based scheduler that uses aggregate utility as an explicit objective function

Questions

- For details on the survey collecting real SDSC users' utility curves:

Lee, Cynthia and Allan Snavely. "On the User-Scheduler Dialogue: Studies of User-Provided Runtime Estimates and Utility Functions." *International Journal of High Performance Computing Applications*, vol. 20, pp.495-506, 2006.

- Contact: Cynthia Lee CL@SDSC.EDU

✓ Shameless plug: I'll be on the academic job market next cycle