dV/dt: Accelerating the Rate of Progress towards Extreme Scale Collaborative Science

Miron Livny (UW)
Bill Allcock (ANL)
Ewa Deelman (USC)
Douglas Thain (ND)
Frank Wuerthwein (UCSD)

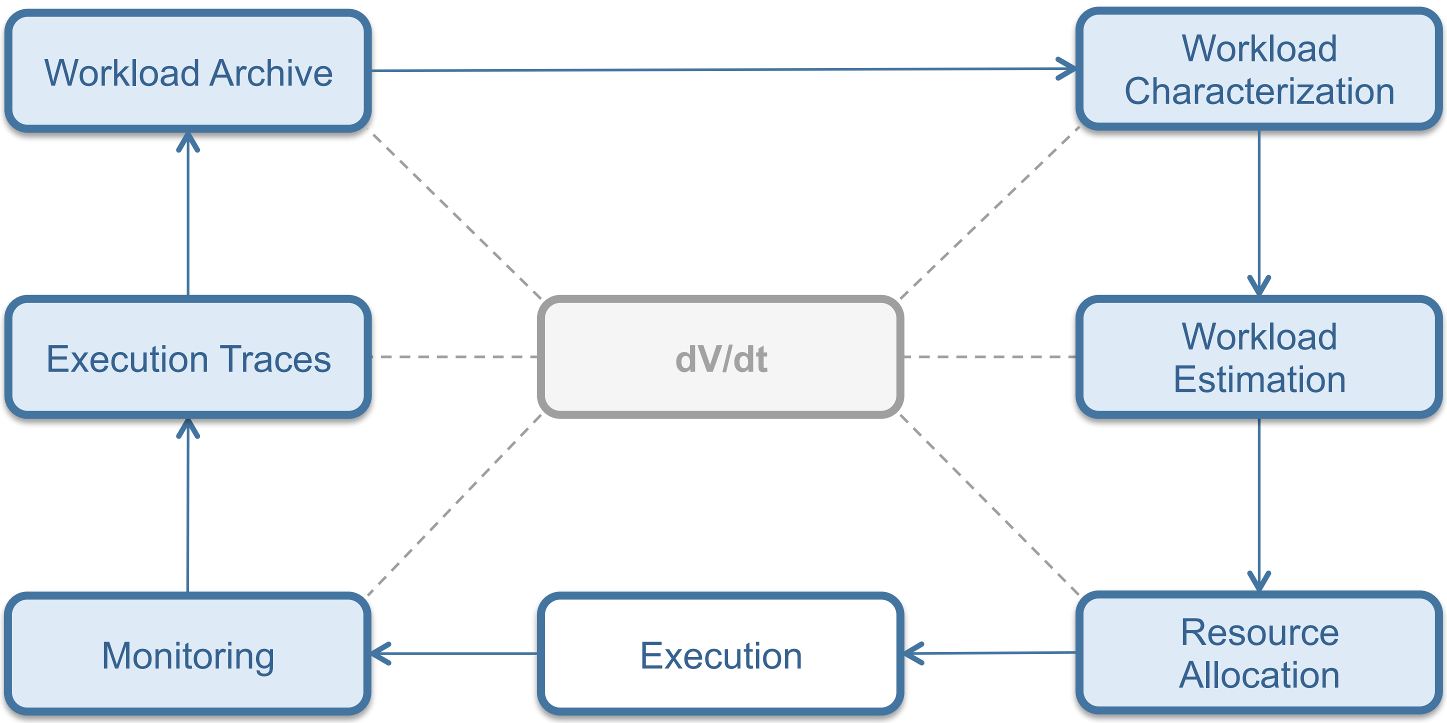
https://sites.google.com/site/acceleratingexascale

USC Viterbi
School of Engineering
Information Sciences Institute

Goal

- “make it easier for scientists to conduct large-scale computational tasks that use the power of computing resources they do not own to process data they did not collect with applications they did not develop”
- In practice: Monitoring, modeling and resource provisioning, scheduling and workload management

Overview of the Resource Provisioning Loop



Monitoring Resource Usage

HTC Monitoring (USC and ND)

- **Job wrappers that collect information about processes**
  - Runtime, peak disk usage, peak memory usage, CPU usage, etc.
- **Mechanisms**
  - Polling (not accurate, low overhead)
  - ptrace() system call interposition (accurate, high overhead)
  - LD_PRELOAD library call interposition (accurate, low overhead)
- **Kickstart (Pegasus) and resource-monitor (Makeflow)**

Error (Accuracy)

	Polling	LD_PRELOAD	Ptrace (fork/exit)	Ptrace (syscalls)
CPU	0.5% - 12%	0.5% - 5%	< 0.2%	< 0.2%
Memory	2% - 14%	< 0.1%	~ 0%	~ 0%
I/O	2% - 20%	0%	0%	0%

Overhead

	Polling	LD_PRELOAD	Ptrace (fork/exit)	Ptrace (syscalls)
CPU	low	low	low	low
Memory	low	medium	low	medium
I/O	low	low	low	high

Gideon Juve, et al., Practical Resource Monitoring for Robust High Throughput Computing, University of Southern California, Technical Report 14-950, 2014.

USC Viterbi School of Engineering
Information Sciences Institute

HPC Monitoring (ALCF)

- Job information from scheduler (Cobalt)
  - Use scheduler data for both scheduler and individual task data
  - Job runtime, number of cores, user estimates, etc.
- I/O using Darshan
  - Instrumentation automatically linked into codes at compile time
  - Captures POSIX I/O, MPI I/O and some HDF5 and NetCDF functions
  - Amount read/written, time in I/O, files accessed, etc.
  - Very low overhead in both time and memory
- Performance Counters using AutoPerf
  - Using built-in hardware performance counters
  - Also enabled at compile time
  - Counters zeroed in MPI_Init, and reported in MPI_Finalize
  - FLOPs, cache misses, etc.
  - Users can take control of performance counters preventing this from working

Workload Modeling and Characterization

USC Viterbi
School of Engineering
Information Sciences Institute

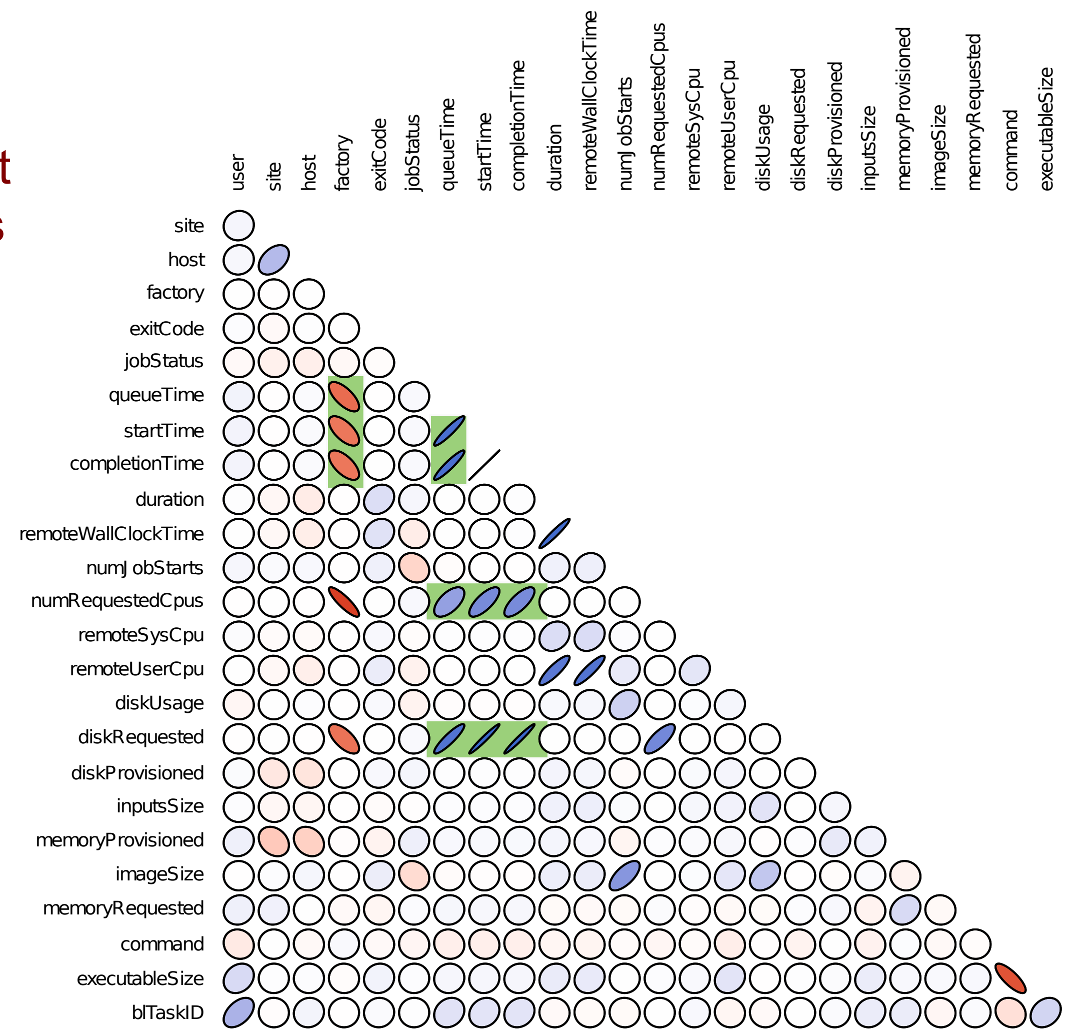
CMS Workload Characteristics (USC, UW-M)

Characteristics of the CMS workload for a period of a month (Aug 2014)

Characteristic	Data
General Workload	
Total number of jobs	1,435,280
Total number of users	392
Total number of execution sites	75
Total number of execution nodes	15,484
Jobs statistics	
Completed jobs	792,603
Preempted jobs	257,230
Exit code (!= 0)	385,447
Average job runtime (in seconds)	9,444.6
Standard deviation of job runtime (in seconds)	14,988.8
Average disk usage (in MB)	55.3
Standard deviation of disk usage (in MB)	219.1
Average memory usage (in MB)	217.1
Standard deviation of memory usage (in MB)	659.6

Workload Characterization

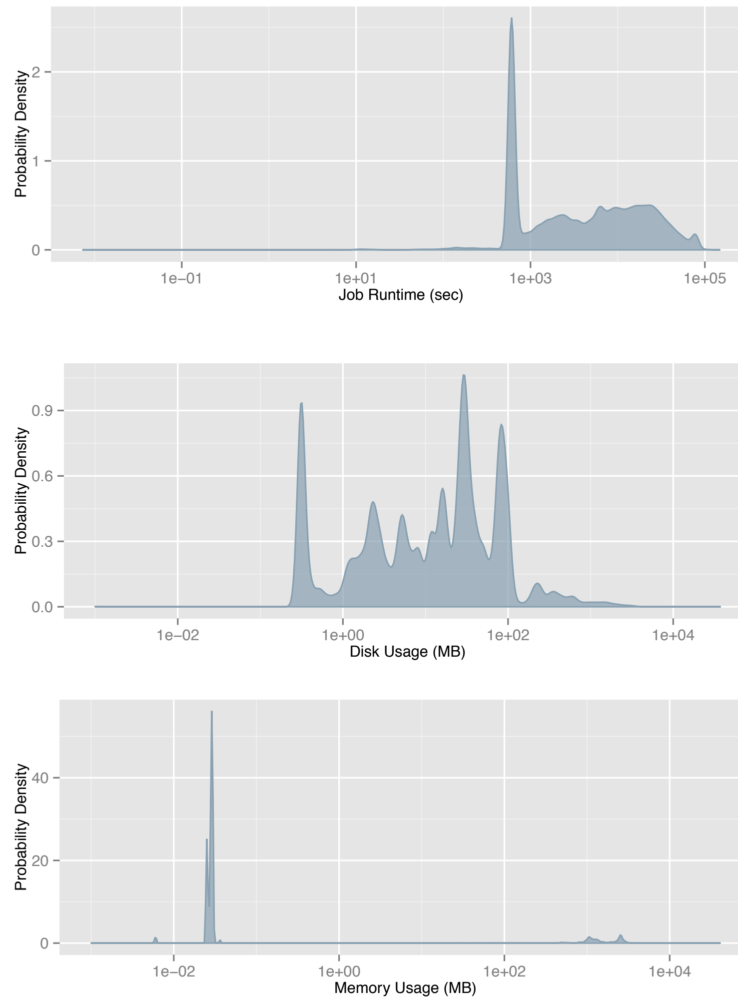
- Correlation Statistics
  - Weak correlations suggest that none of the properties can be directly used to predict future workload behaviors
  - Two variables are correlated if the ellipse is too narrow as a line



USC Viterbi School of Engineering
Information Sciences Institute

Workload Characterization (2)

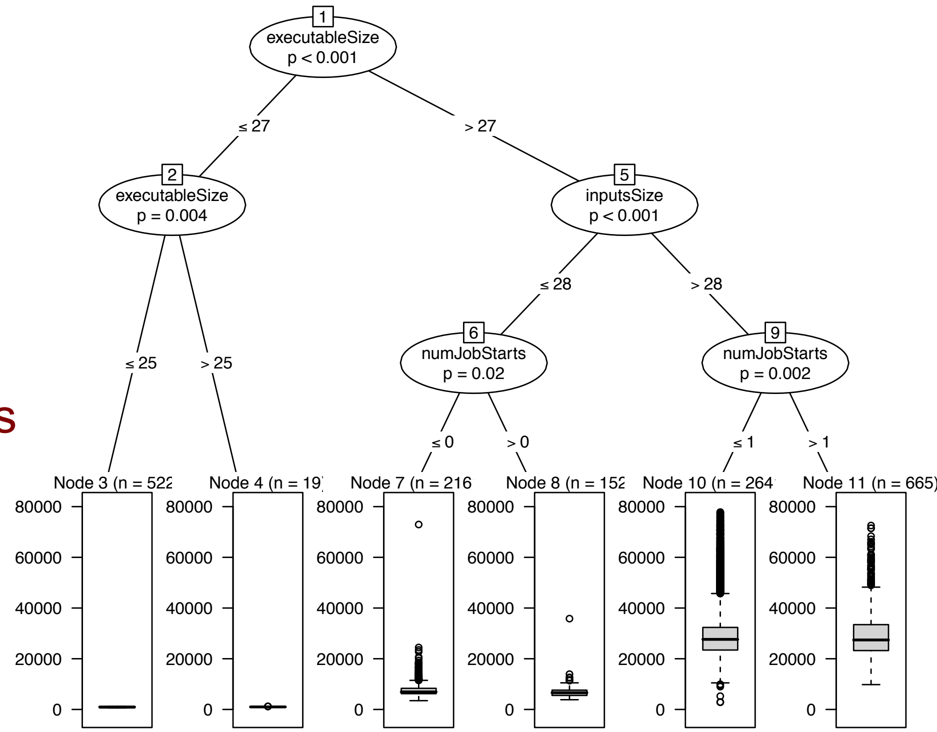
- Correlation measures are sensitive to the data distribution
- Probability Density Functions
  - Do not fit any of the most common families of density families (e.g. Normal or Gamma)
- Our approach
  - Recursive partitioning method to combine properties from the workload to build Regression Trees



USC Viterbi
School of Engineering
Information Sciences Institute

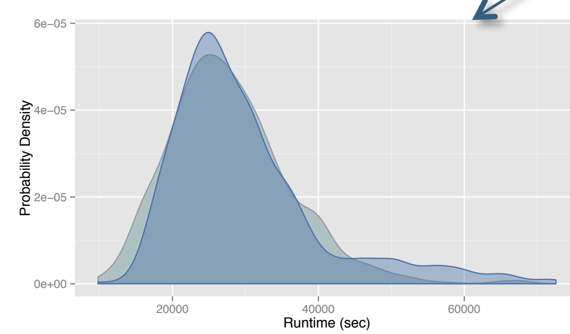
Regression Trees

- The recursive algorithm looks for PDFs that fit a family of density
  - In this work, we consider the Normal and Gamma distributions
  - Measured with the Kolmogorov-Smirnov test (K-S test)



The PDF for the tree node (in blue) fits a Gamma distribution (in grey) with the following parameters:

Shape parameter = 12
Rate parameter = $5x10^{-4}$
Mean = 27414.8
p-value = 0.17



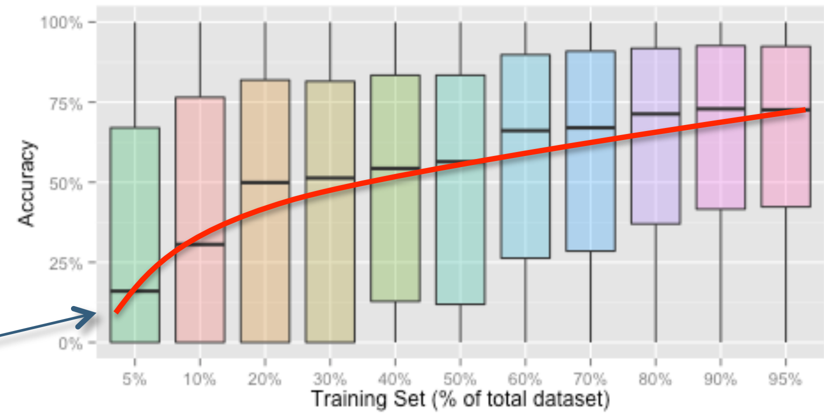
USC Viterbi
School of Engineering
Information Sciences Institute

Job Estimation: Experimental Results

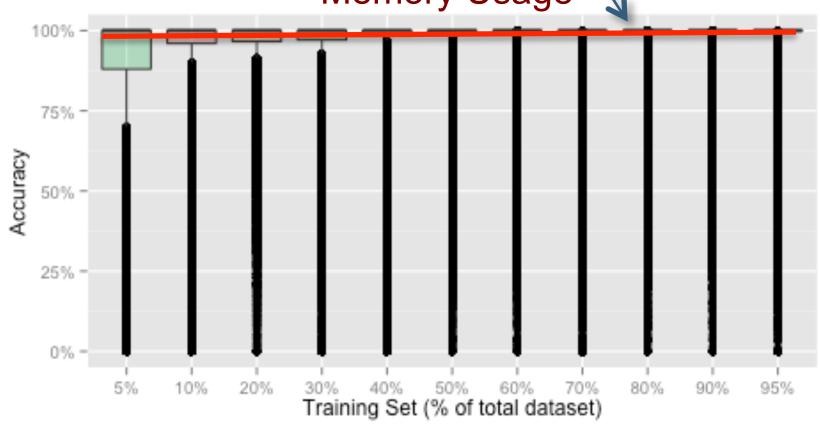
- Based on the regression trees
  - We built a regression tree per user
  - Estimates are generated according to a distribution (Normal or Gamma) or a uniform distribution

The median accuracy increases as more data is used for the training set

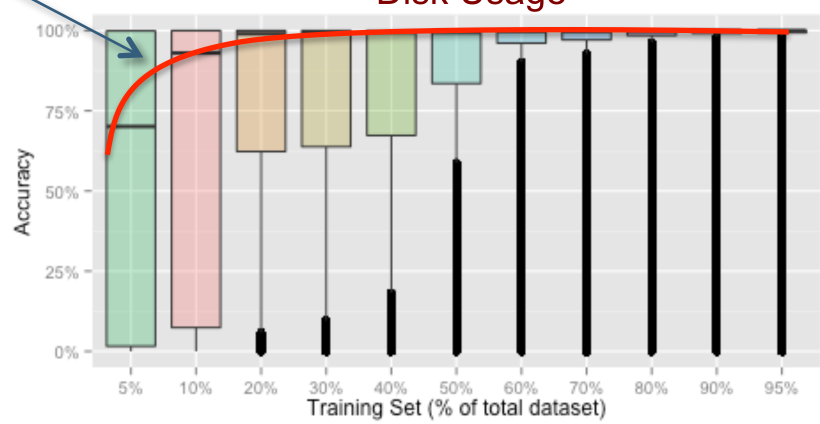
Job Runtime



Memory Usage



Disk Usage



Average accuracy of the workload dataset
The training set is defined as a portion of the entire workload dataset

USC Viterbi School of Engineering
Information Sciences Institute

Provisioning and Resource Allocation

Resource Allocation (ND)

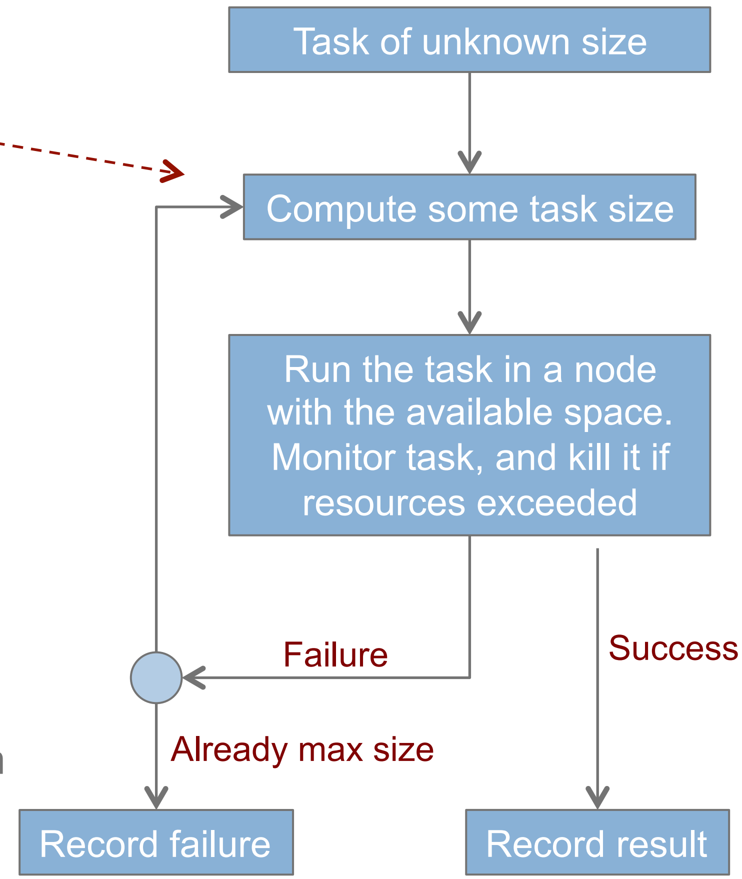
- Tasks have different sizes (known at runtime) while computation nodes have fixed sizes

- Resource allocation strategies
  - One task per node
    - Resources are underutilized
    - Throughput is reduced
  - Many tasks per node
    - Resources are exhausted
    - Jobs fail
    - Throughput is reduced

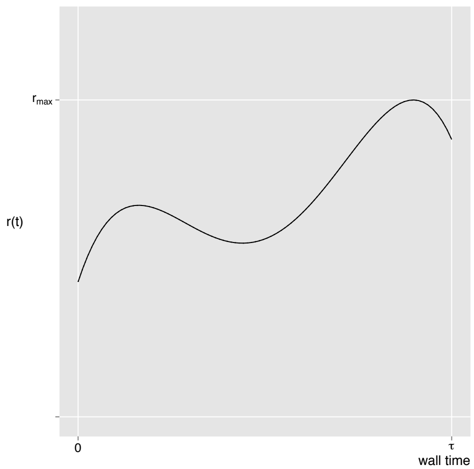
USC Viterbi School of Engineering *Information Sciences Institute*

General Approach

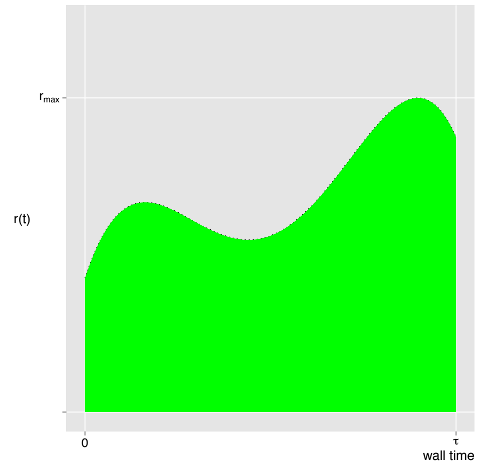
- Setting tasks
  - What do we know?
    - Maximum size?
    - Size probability distribution?
    - Empirical distribution?
    - Perfect information?
- Our approach
  - Setting task sizes to reduce resource waste
    - Modeling of resource sizes (e.g., memory, disk, or network bandwidth)
    - Assumes the task size distribution is known
    - Adapts to empirical distributions



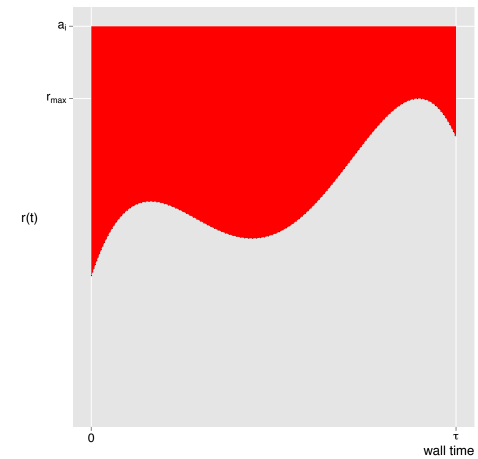
Resource Waste Modeling



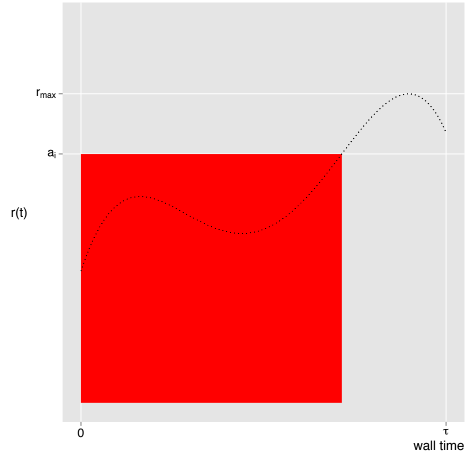
Model the task resource
as a function of time



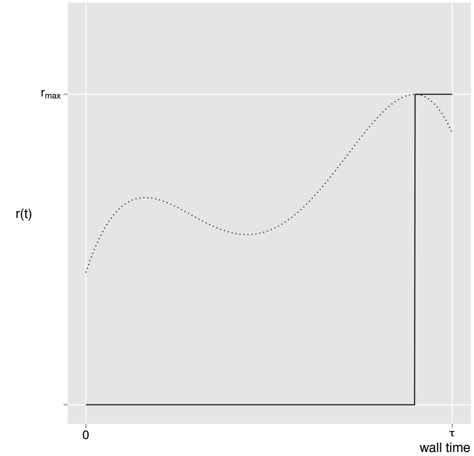
Model the task resource usage as
resource x time (area below the curve)



Overestimating size
(waste is the area above the curve)



Underestimating size
(waste is resource x time
until resource exhaustion)

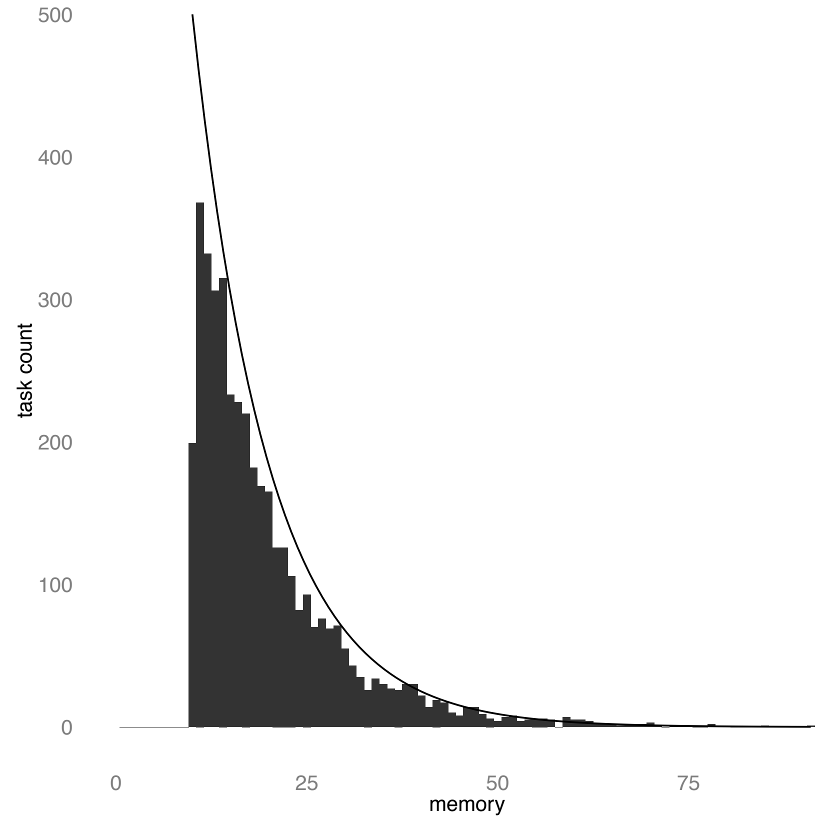


Single Peaks Model
Simplifying assumption: any resource exhaustion
only happens at time of maximum peak
(i.e., resource usage looks like a step function)

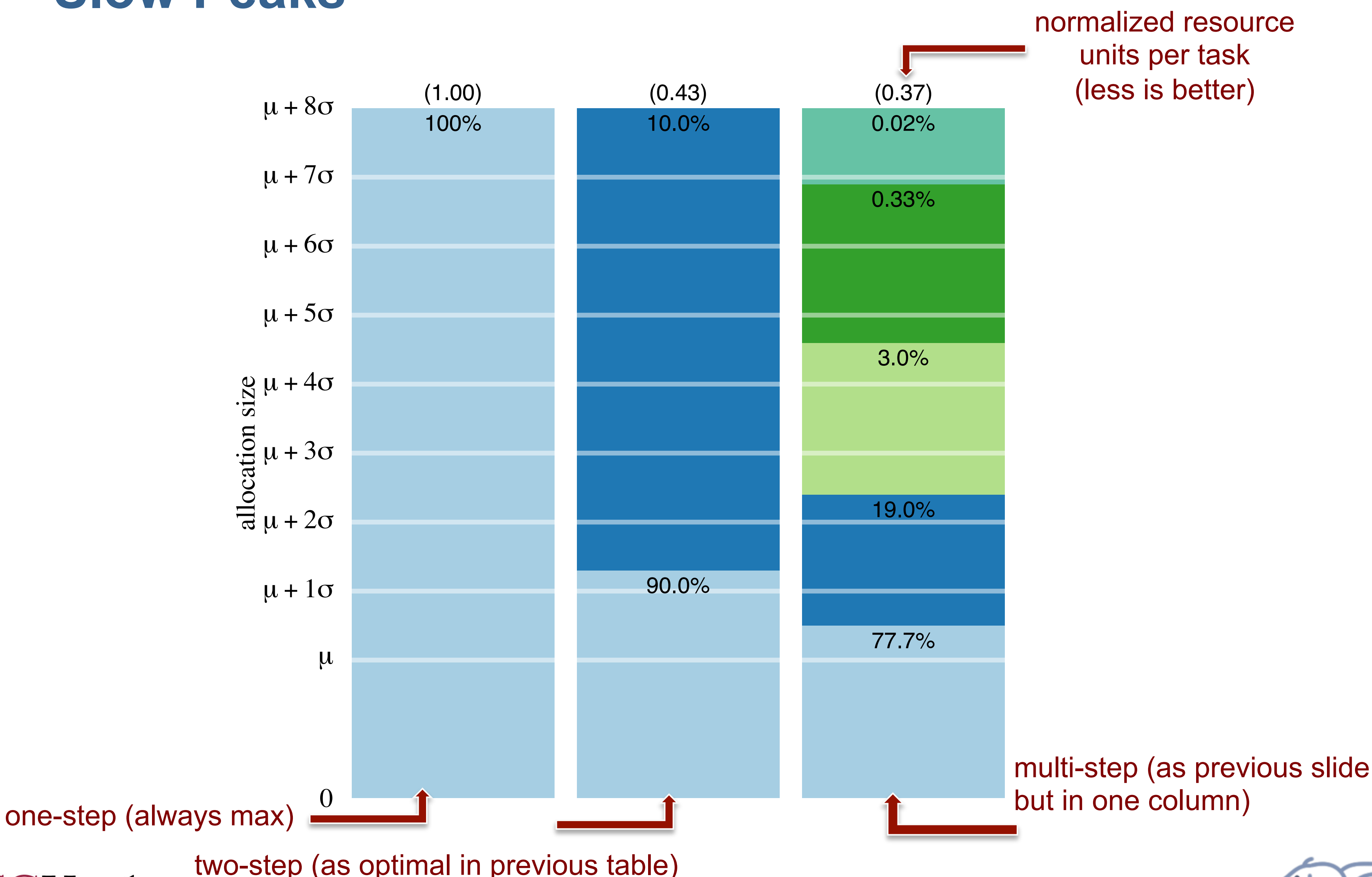
USC Viterbi
School of Engineering
Information Sciences Institute

Synthetic Workload Experiment

- Exponential Distribution
  - 5000 Tasks
  - Memory according to an exponential distribution
    - Shifted min 10 MB, truncated max 100 MB, average 20 MB
  - Tasks run anywhere from 10 to 20 seconds
  - 100 computation nodes available, from ND Condor pool
  - Each node with 4 cores and a limit of 100 MB of memory



Example: One,Two and Multi-step sequences with "Slow Peaks"



Next Steps

- Improve monitoring and modeling
  - Investigate network I/O and energy
  - Extend modeling to parallel, HPC applications
- Close the loop
  - Turn on detailed monitoring in workflows
  - Use resource predictions for provisioning and scheduling
- Productize tools
  - Deploy monitoring capabilities in production environments
  - Turn modeling software into a service