Generating Reliable Predictions of Batch Queue Wait Time

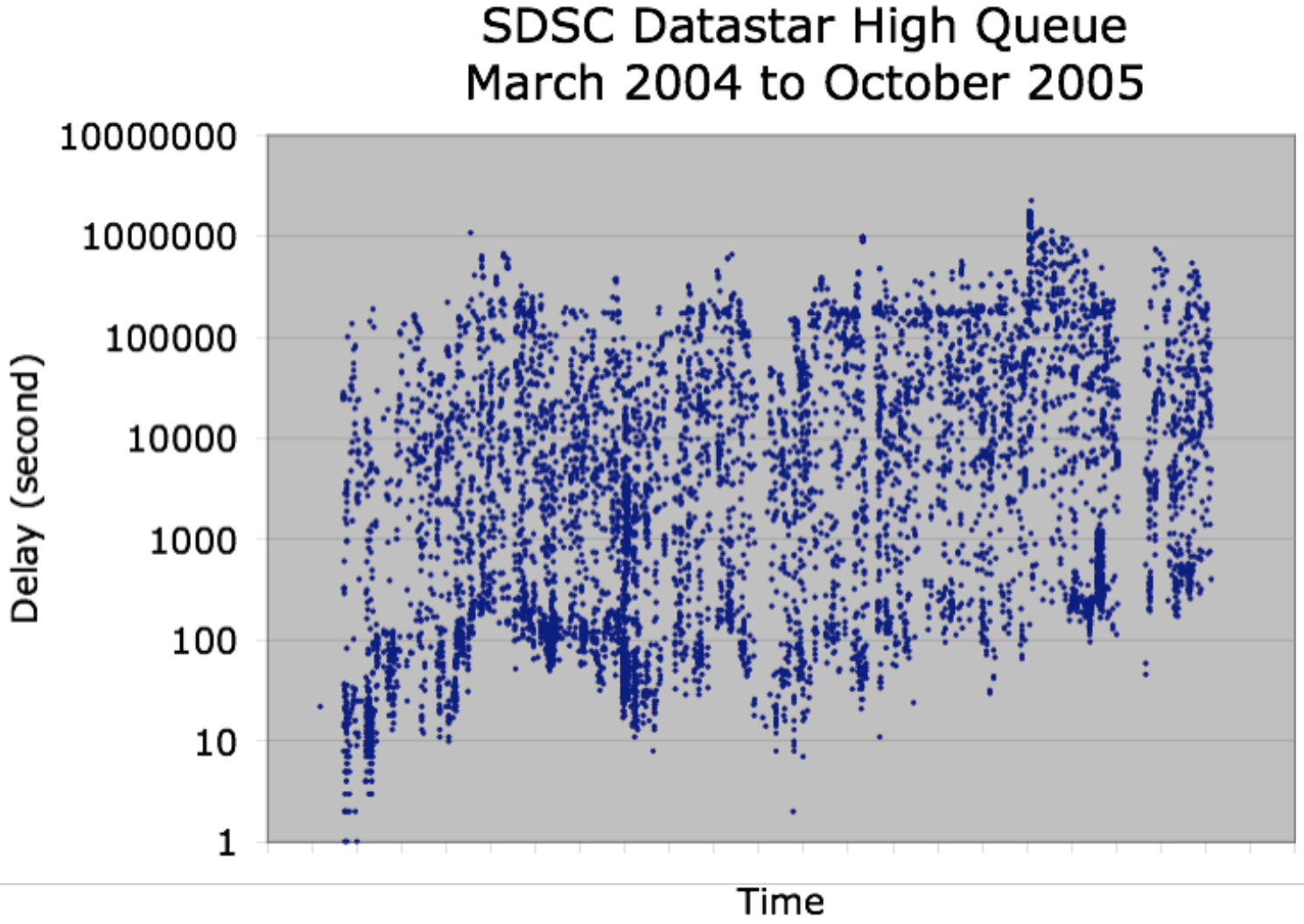
Rich Wolski

How Long will my Job Wait in the Queue?

- This is an old problem
  - LLNL circa 1986
- Well studied
  - JSSPP in its 20th year

- Most methodologies are based on the idea of making a prediction that is correct “on the average.”
  - “The expected value”

Here's the Problem with that Approach



Thought Experiment

- Imagine 100 jobs, in random order, that experience 6 orders of magnitude variation in queue wait delay
  - 94 jobs wait 10 seconds
  - 1 job waits 100 seconds
  - 1 job waits 1000 seconds
  - 1 job waits 10000 seconds
  - 1 job waits 100000 seconds
  - 1 job waits 1000000 seconds
  - 1 job waits 10000000 seconds
- ***What prediction would you make about the next job's wait time?***
- ***"On the average"***, the wait time is 111120 seconds
- ***Rich's prediction:*** The next job will wait 10 seconds with probability 0.94

Predicting using Quantiles

- Provide the user with an upper bound on wait time associated with a specific probability
  - ***Your job will wait no more than 10 seconds with probability 0.94***
- So? Why did this take 20 years?
  - The average is a good estimator of the mean in most cases
  - What is a good estimator of an arbitrary quantile?
  - No explanatory power

Simple but Ineffective

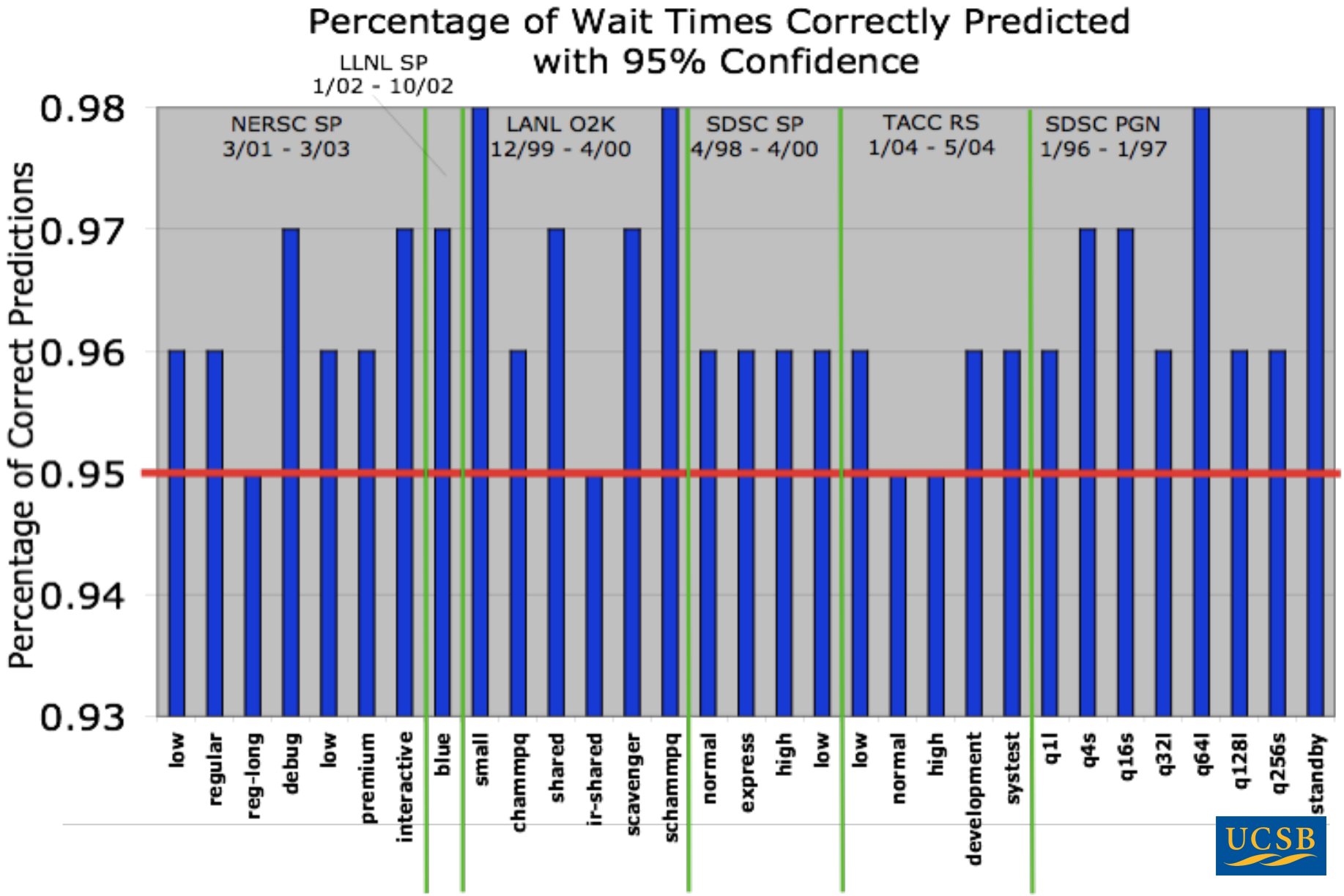
- Sort the wait times into a list of size **s** and for probability **p** take the **(p * s)th** largest value in the sorted list
- In the previous example
  - Sort all 100 jobs by wait time
  - For **p = 0.94**, 94th largest value is 0.94 quantile
- Why does this approach not work?
  - Unless sample size is large, the chances of seeing a value close to the quantile are small with the quantiles are near the tails
  - Autocorrelation: for queue wait times, the order matters

Quantile Bounds Estimation from Time Series

- Uses Binomial model to determine upper or lower bound on a specific quantile from empirical data
  - Requires a confidence bound **C** to be specified for the quantile of interest
  - **C** determines the probability that the QBETS bound is "wrong"
  - The QBETS bound is conservative according to **C**
    - If you want QBETS to be really really right about the upper bound (C = 0.9999), then it will guess a large number so that probability of being below the "true" quantile is 0.0001
- For Supercomputer Batch queues, **C=0.95** works pretty well
- Also manages autocorrelation

UCSB

How well?



UCSB

Clustering

- Most batch schedulers use job characteristics to determine dynamic priorities within each queue
- For example: backfilling
  - Large job (many nodes) in the queue
  - Small jobs that will run before the large job gets to the head of the queue “jump ahead”
  - Requires node count and maximum run time for each job to be visible to the scheduler
- Should be able to improve QBETS conservativeness by clustering jobs with similar characteristics

When Dinosaurs Roamed the Earth

- DataStar Normal Queue Oct 27, 2005, 4:34 PM PT
  - 1 to 16 processors: 478358s (5.5 days)
  - 17 to 170 processors: 644053s (7.5 days)
- University of Chicago TG System, Oct 27, 2005, 4:34 PM PT
  - 1 processor: 7557s (2.1 hours)
  - 2 to 92 processors: 29684s (8.24 hours)

And after the Invention of FaceBook

- TACC Stampede, June 29, 2016, 8:51 PM
  - 1 to 6 processors: 84675s (~1 days)
  - 7 and 8 processors: 172093s (~2 days)
  - 9 to 256 processors: 126102s (~1.5 days)
- SDSC Comet, June 29, 2016, 8:51 PM
  - 1 to 72 processors: 86245s (~1 day)

Right Answer, Wrong Question

- QBETS takes a probability and confidence (**p** = 0.95, **c** = 0.95) and returns a time
  - *What is the time associated with this probability?*
- What is the probability that my job will start by a certain time?
  - *What is the probability associated with this time?* => Inverse QBETS
- For example: ***What is the probability that a 32 node job, that will run for 30 minutes will start in the next 24 hours?***
- June 29, 2016 -- Stampede: **0.95**, Comet: **0.75**
  - For comparison – Oct 27, 2005 Datastar: **0.8**
- Computationally much harder

UCSB

Current Status

- The core prediction code is still active
  - Aristotle is using it to predict prices in the AWS spot market
- Web site code use for TG (QBETS and Inverted QBETS) and the data collectors are no longer available
- To see production deployment
  - Need a new web service
  - Need to ensure data acquisition is correct
  - Performance tuning for on-line response

The RACE Lab

- Shared with Chandra Krintz
  - Kyle Carson
  - Stratos Dimopoulos
  - Nevena Golubovic
  - Hiranya Jayathilaka
  - Benji Lampel
  - Kevin Malta
  - Alex Pucher
- **http://www.cs.ucsb.edu/~ckrintz/racelab.html**
- **rich@cs.ucsb.edu**
- **ckrintz@cs.ucsb.edu**

UCSB