Predicting Queue Times on Space-Sharing Parallel Computers

Allen B. Downey

University of California at Berkeley

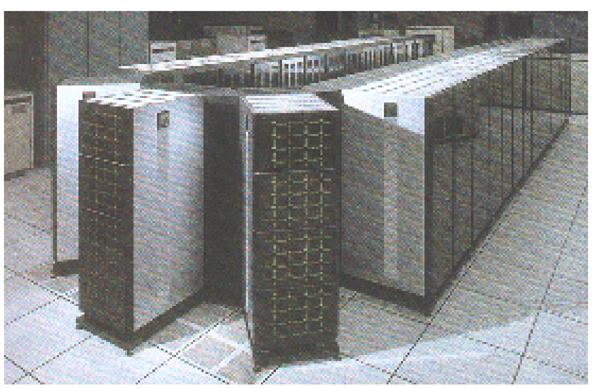
San Diego Supercomputer Center

Motivating problem

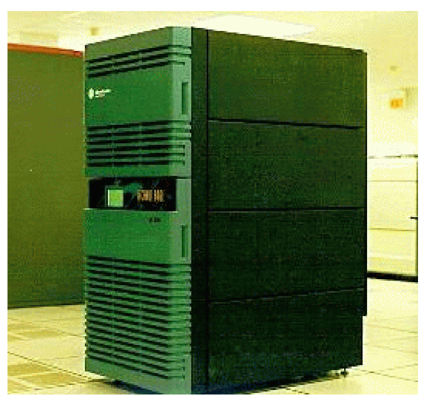
External resource selection:

- Choosing among computing resources.
- Predict queue time from arrival until beginning of execution.
- Predict run time.
    - Application characteristics
    - Quality-of-service forecasts

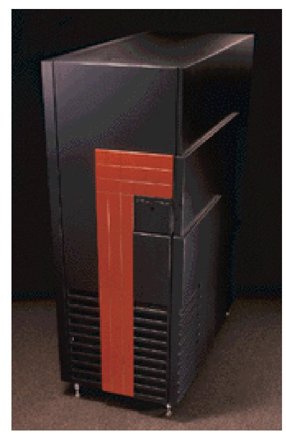
AppLeS



eeny



meeny



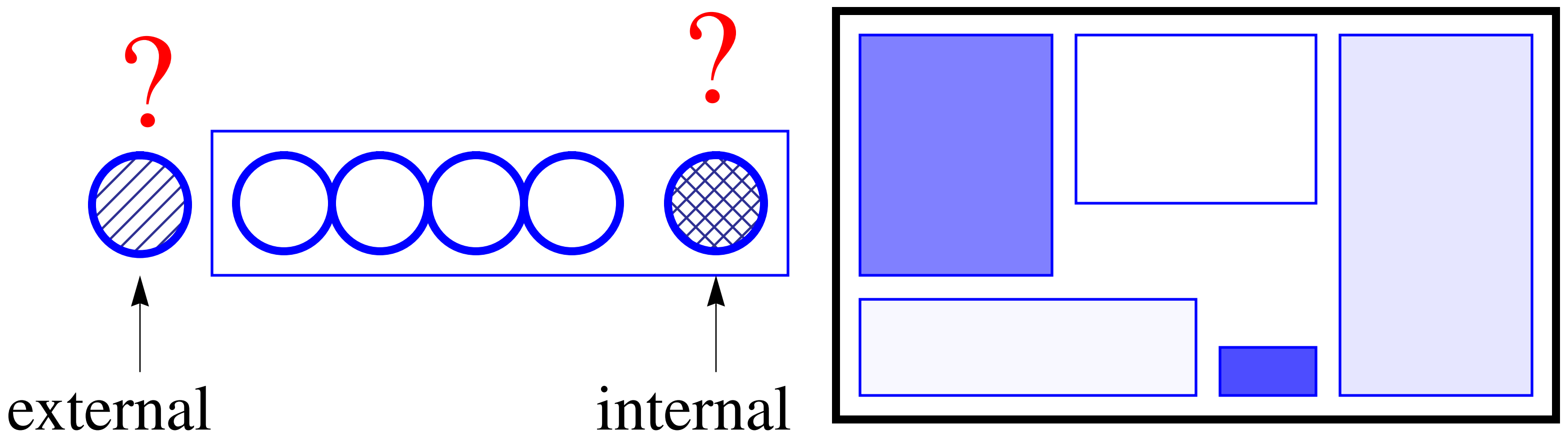
miny

mo

Other motivating problem

Internal resource selection:

- Choosing number of processors for malleable jobs.
    - Range of cluster sizes.
    - Cannot change cluster size dynamically.
- Predict queue time until n processors are available.



Outline

- Motivating problems
- Lifetime distributions $\longrightarrow$ predict individual jobs
- Individual predictions $\longrightarrow$ predict queue times
    - Predictor A, median
    - Predictor B, mean
- Results
- Future work

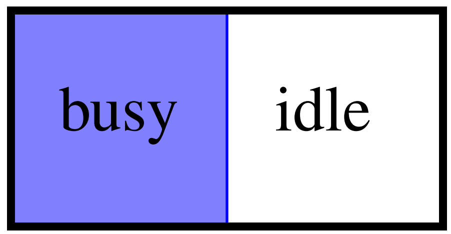
Example

R(n): run time as a function of cluster size

Q(n): predicted queue time until n processors are free

R(1) = 60 min.

R(2) = 40 min.

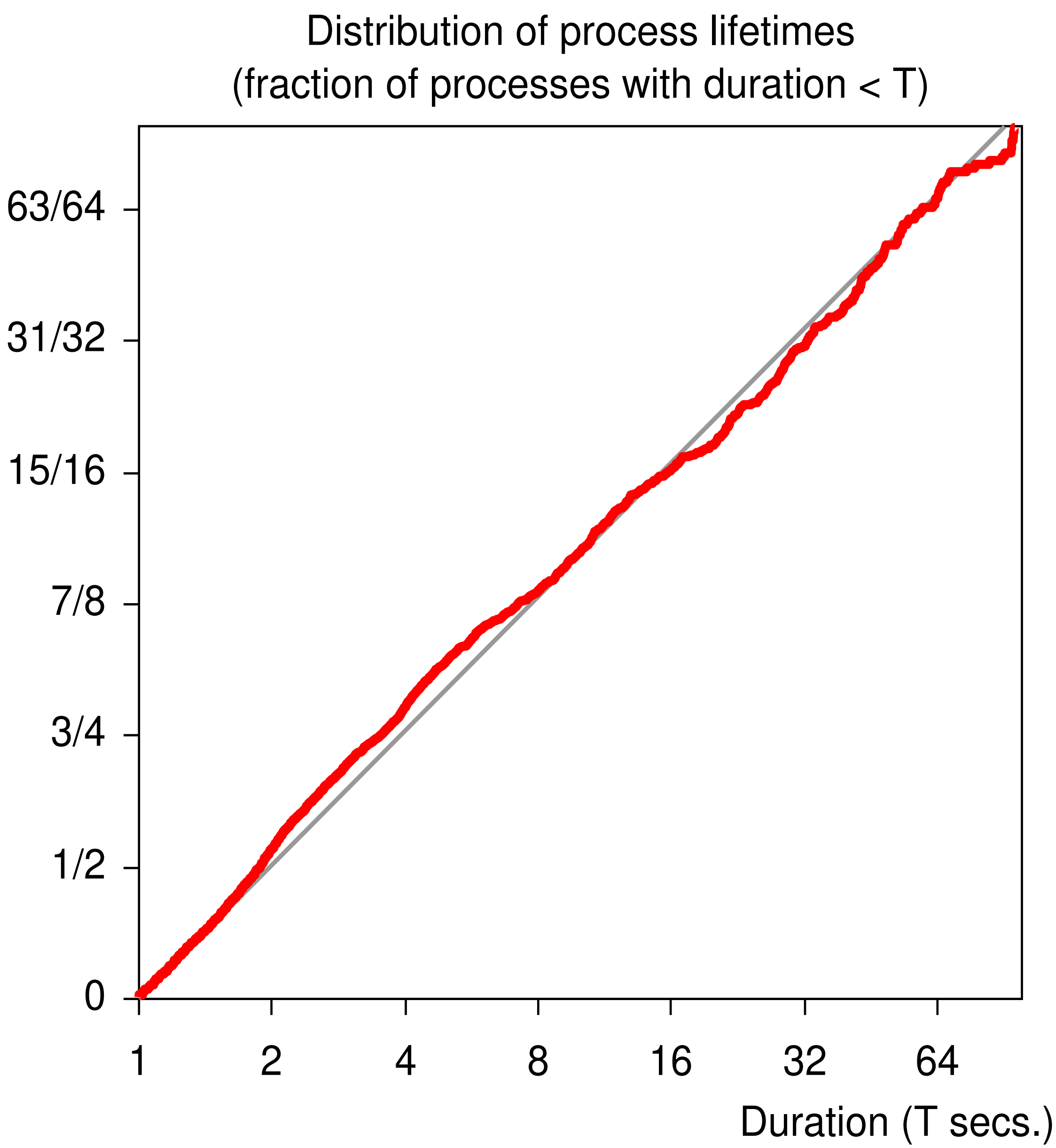


Q(1) = 0

Q(2) = ???

- Need to predict remaining lifetime of running job.
- Best indicator: age
- Past repeats?

Distribution of lifetimes: Pareto



- Interactive UNIX processes on workstations.
- Median remaining lifetime = age.

[LelandOtt86]

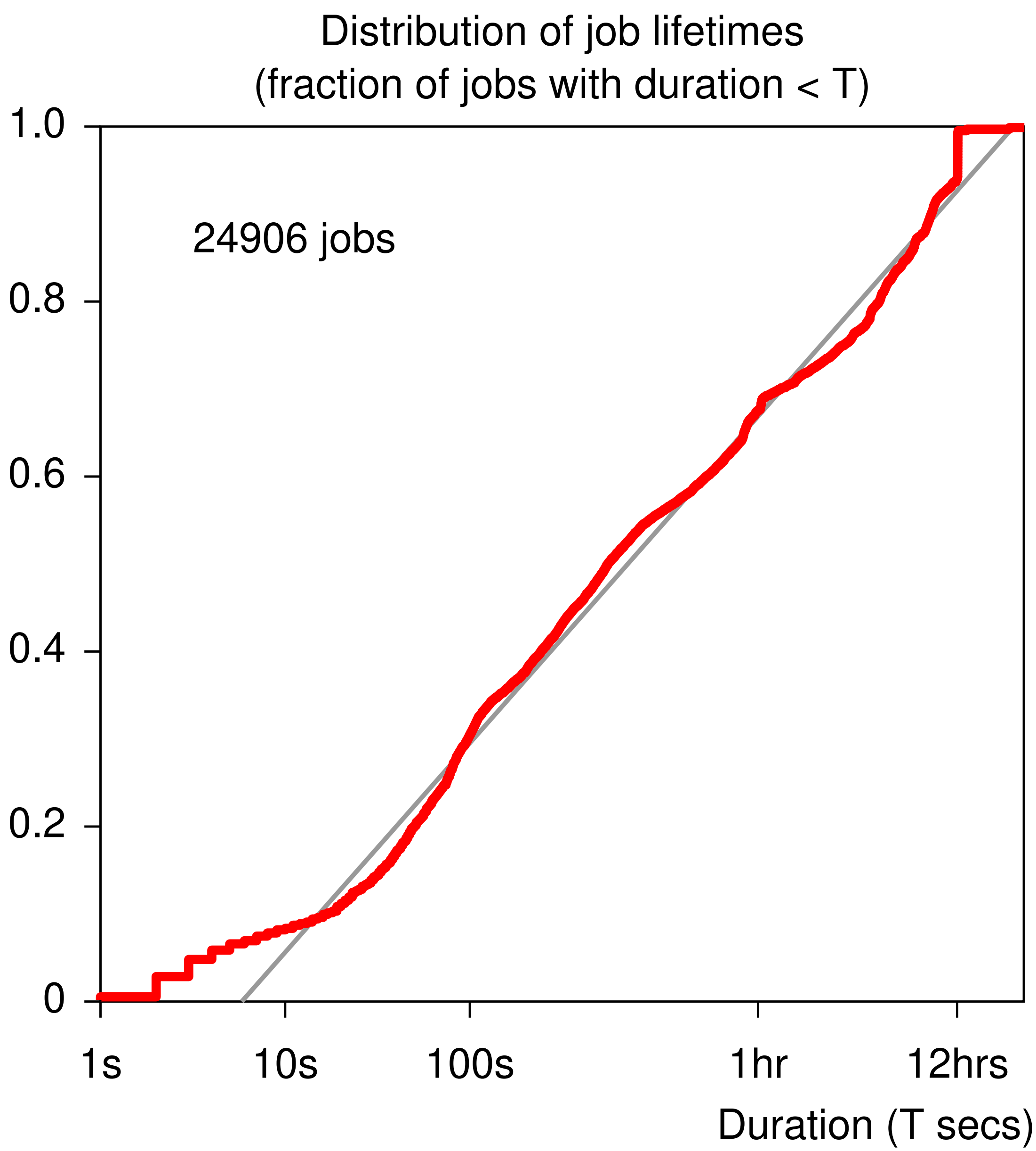
[Harchol-BalterDowney96]

Conditional distributions

Distribution:	Remaining lifetime:
Pareto	increases linearly with age (UBNE).
Uniform	decreases linearly (NBUE).
Exponential	constant.
etc.	*etc.*

Can't predict remaining lifetime
unless we know the distribution.

Distribution of lifetimes: uniform-log

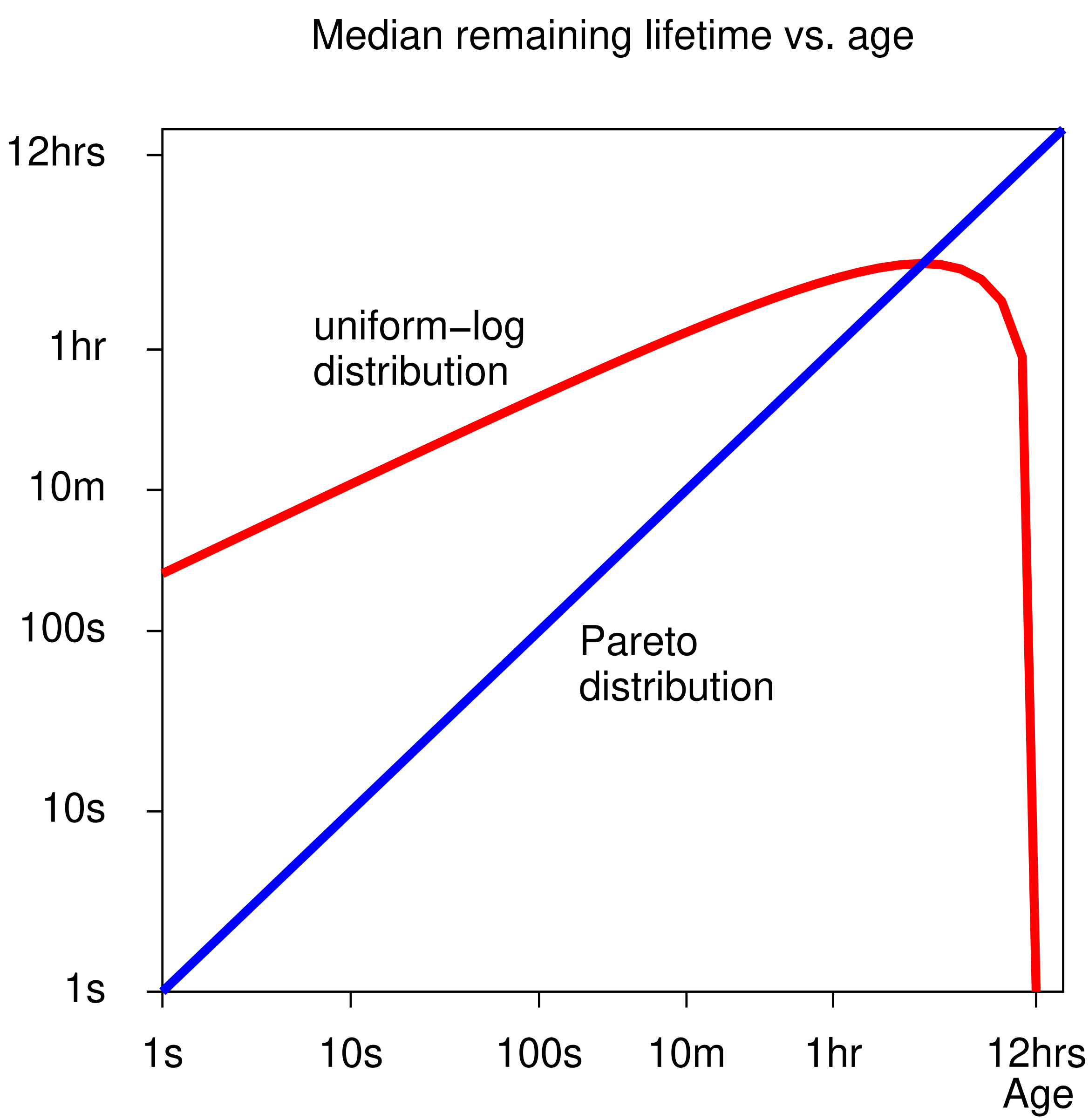


- Batch jobs on Paragon at SDSC.
- Same shape on:

  Cray C90 at SDSC

  IBM SP2 at CTC.

Remaining lifetime vs. age

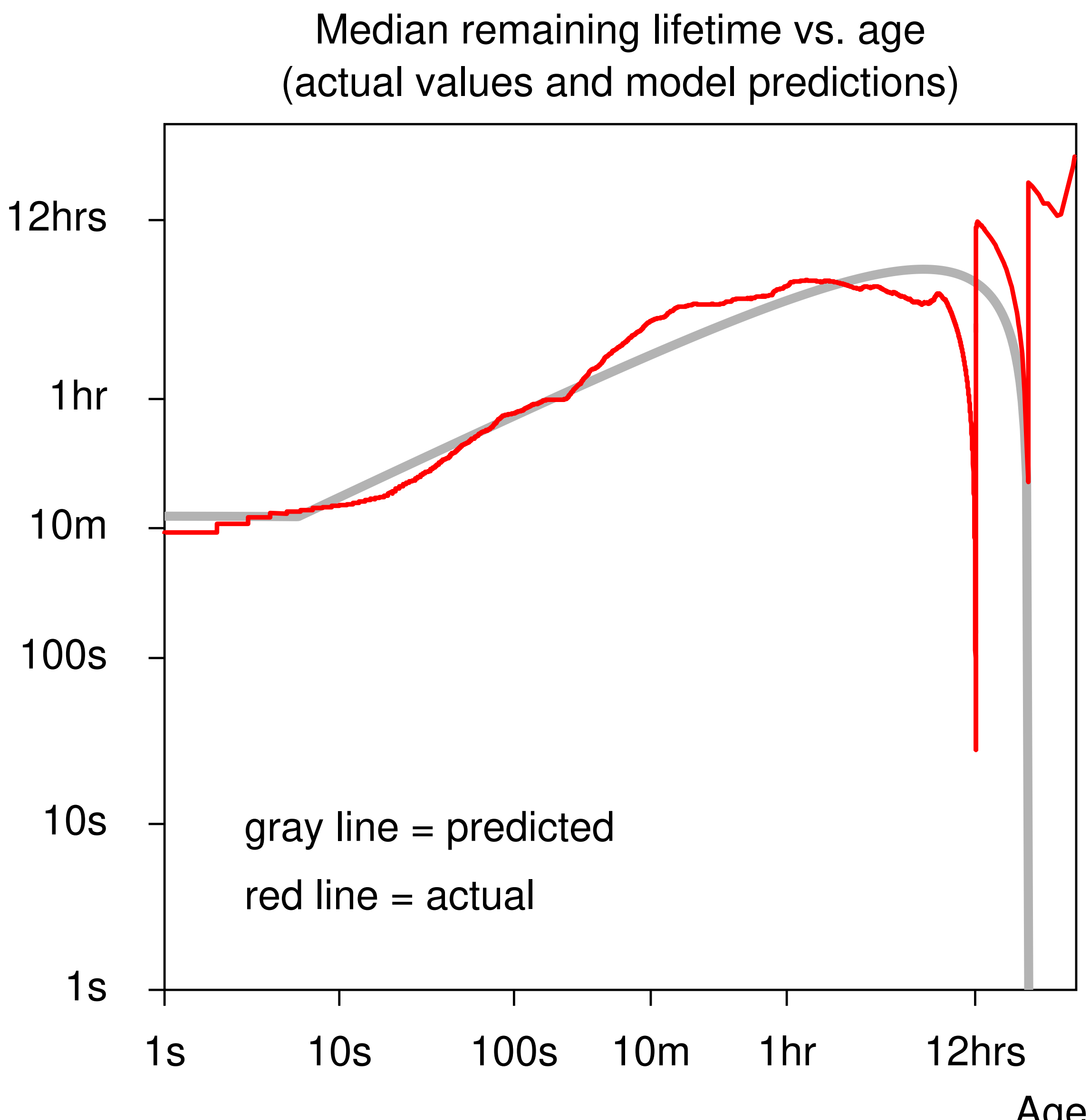


- Median remaining lifetime =

$$\boxed{\sqrt{t_m a} - a}$$

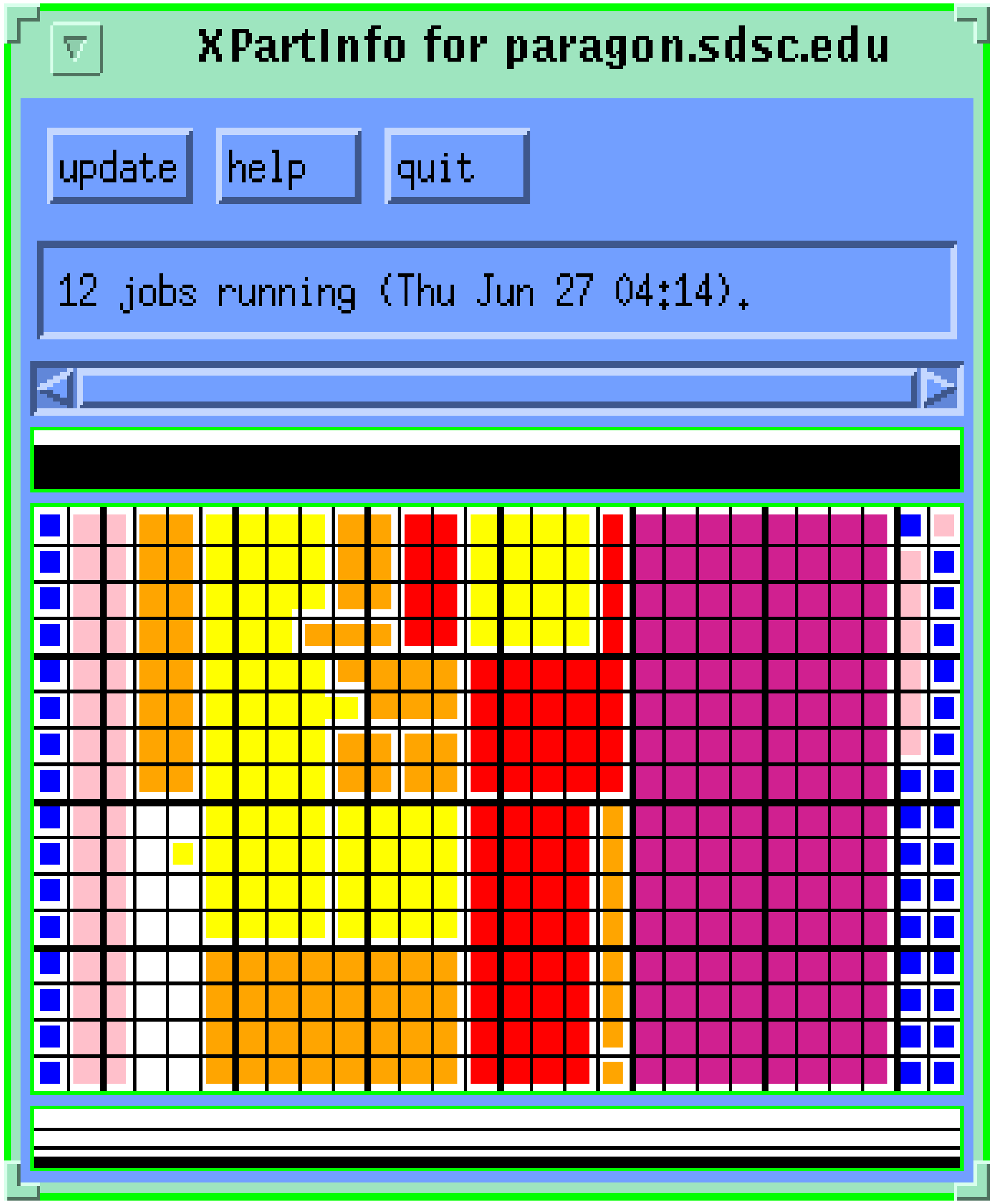
- Increases slowly and drops off eventually.

Remaining lifetimes: a dose of reality



- Good agreement up to time limit.
- Distribution of very old jobs unknown.
- Some special cases—not critical, though.

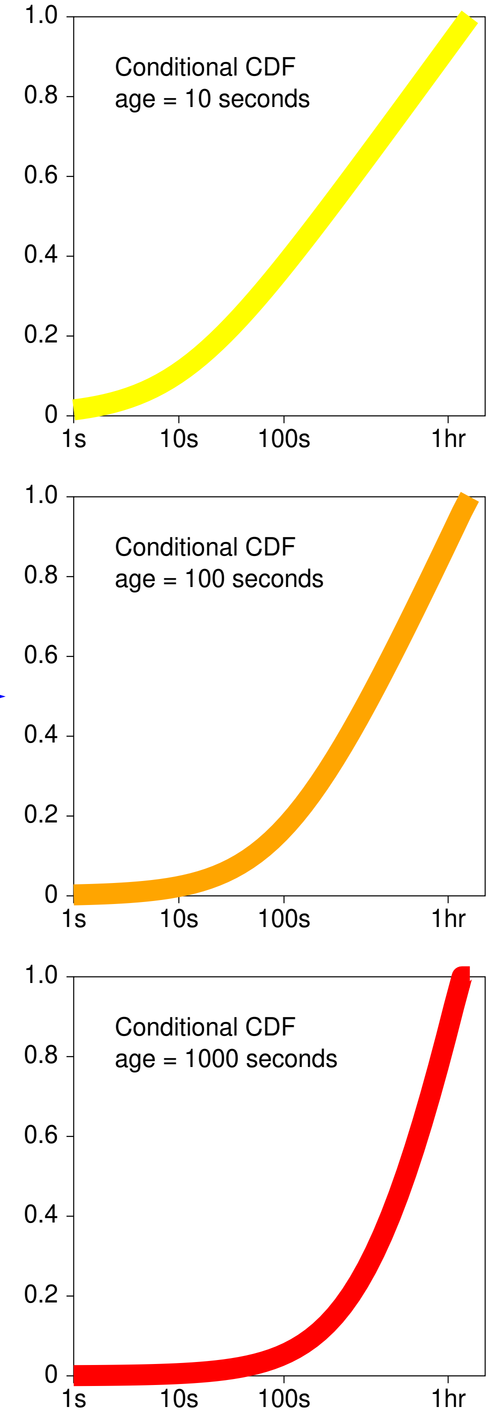
Considering system state



- Number of free nodes
- Number of running jobs
- Cluster size of each running job
- Ages of running jobs
- Class of running jobs (e.g. queue names)

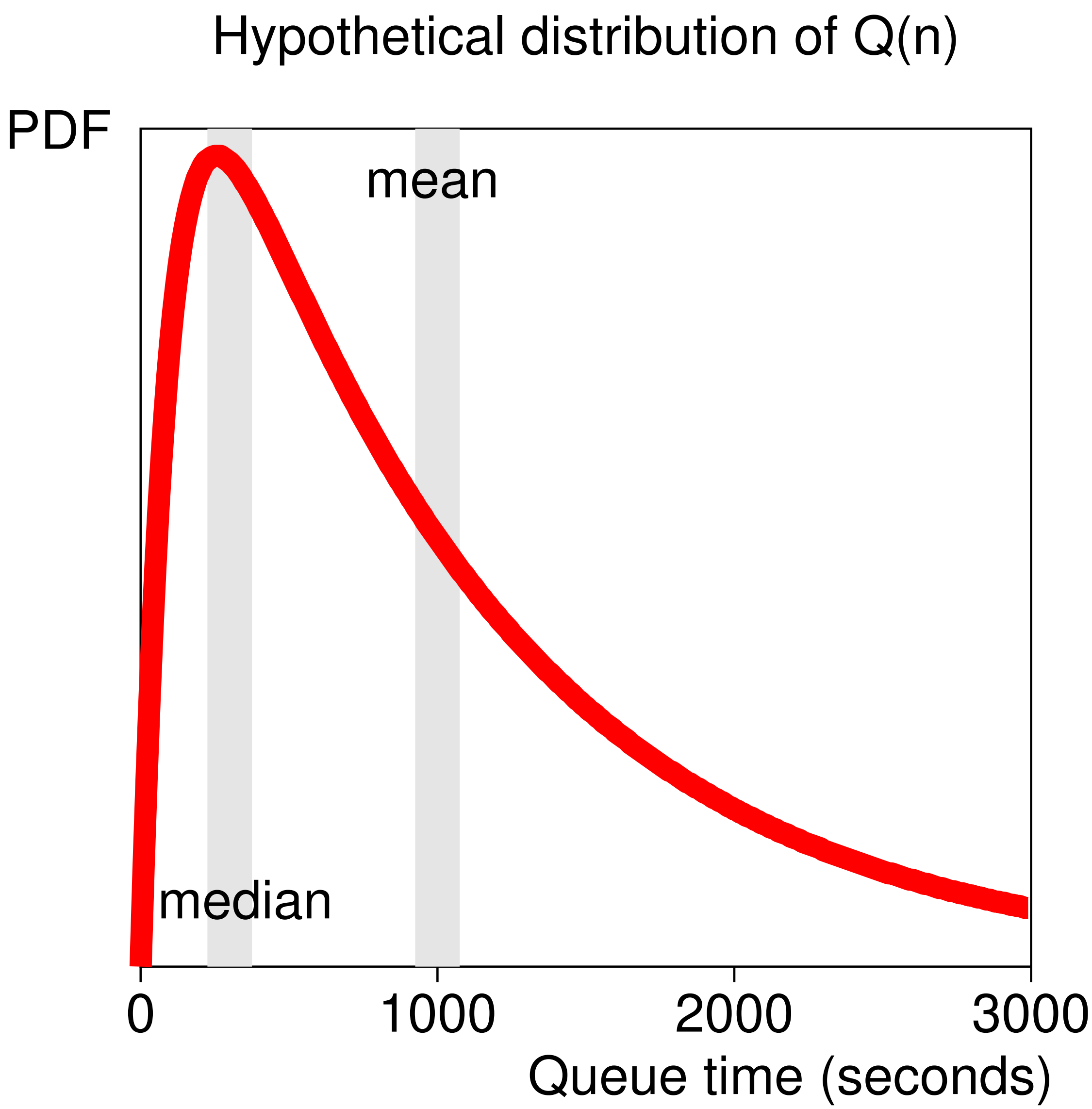
The Hard Way

Job	age	#pes
1	10	16
2	100	8
3	1000	32



Distribution of
$Q(n) =$

Making predictions

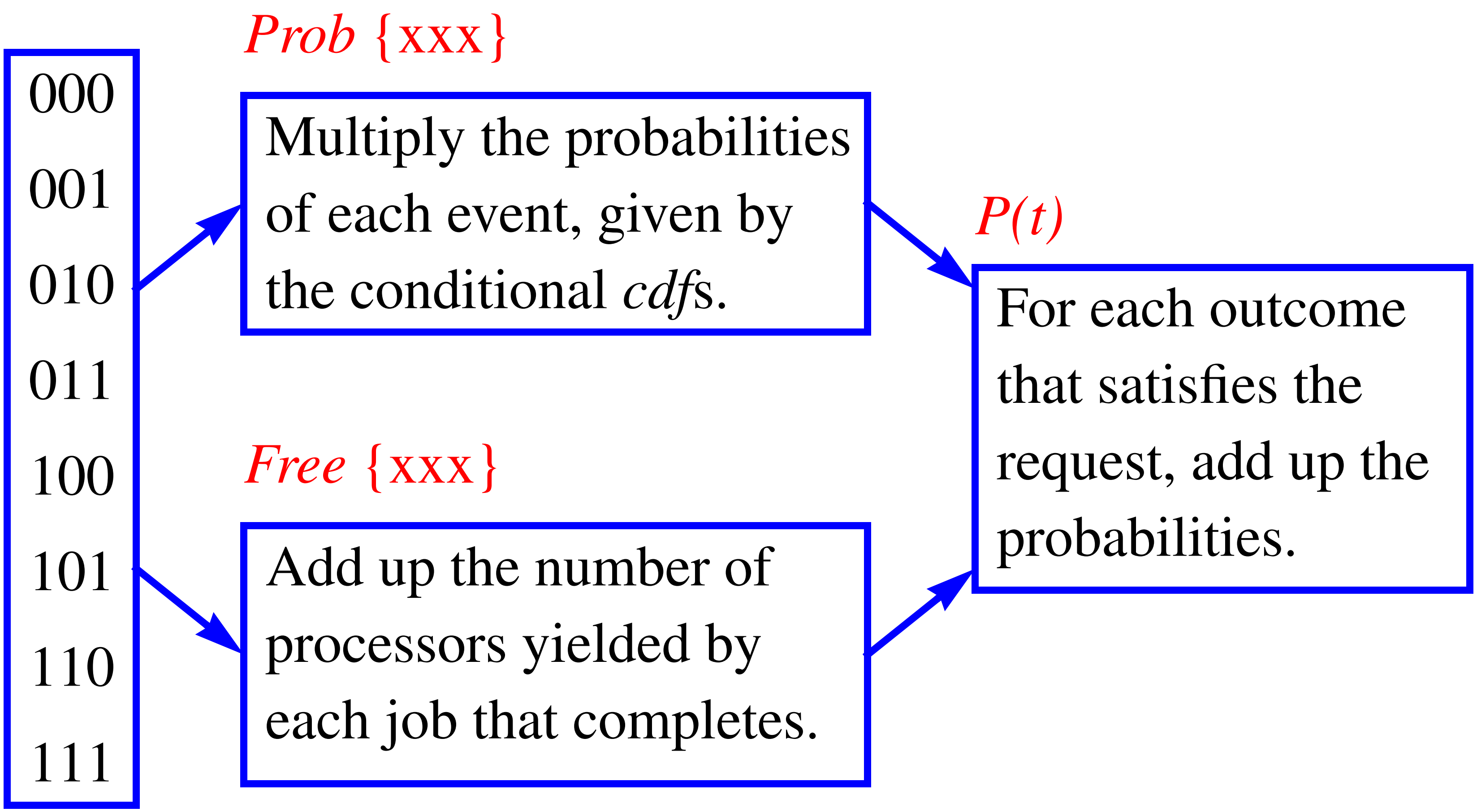


- Don't need whole distribution.
- Mean minimizes mean-square errors.
- Median gets most hits.
- Distribution of *Q(n)* is often symmetric.

Predictor A (exact)

Estimate median of $Q(n)$: solve $P(t) = \text{Prob}\{\begin{smallmatrix}\text{request satisfied}\\ \text{before time t}\end{smallmatrix}\} = 0.5$

For each possible outcome:



Predictor A (approximate)

Estimate median of *Q(n)*.

Identify potential benefactors, and group potential outcomes.

Example: $n = 12$

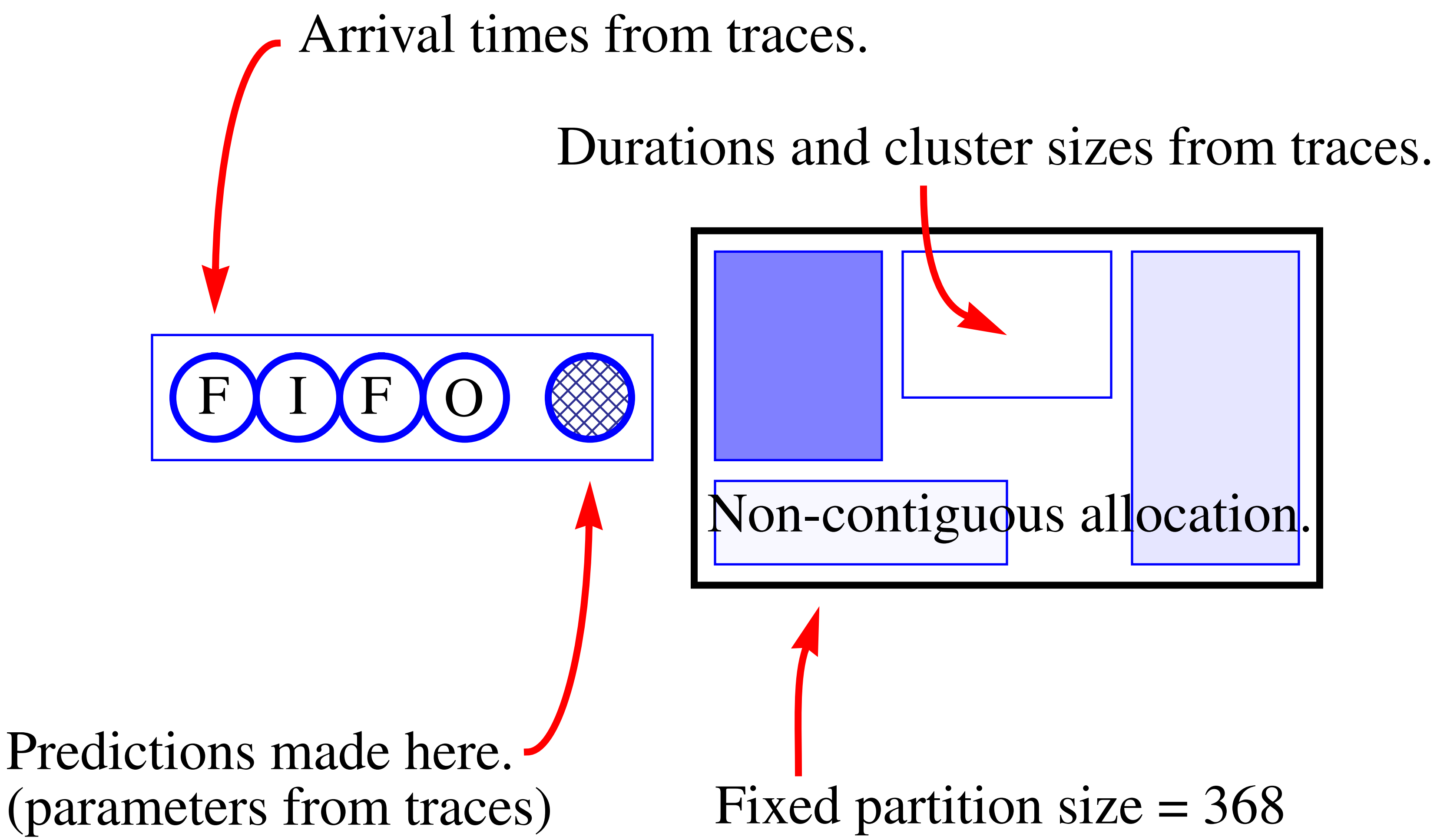
Job	age	#pes
1	10	16
2	100	8
3	1000	32

Probability that none of the benefactors completes is roughly the probability that the request is not satisfied before t:

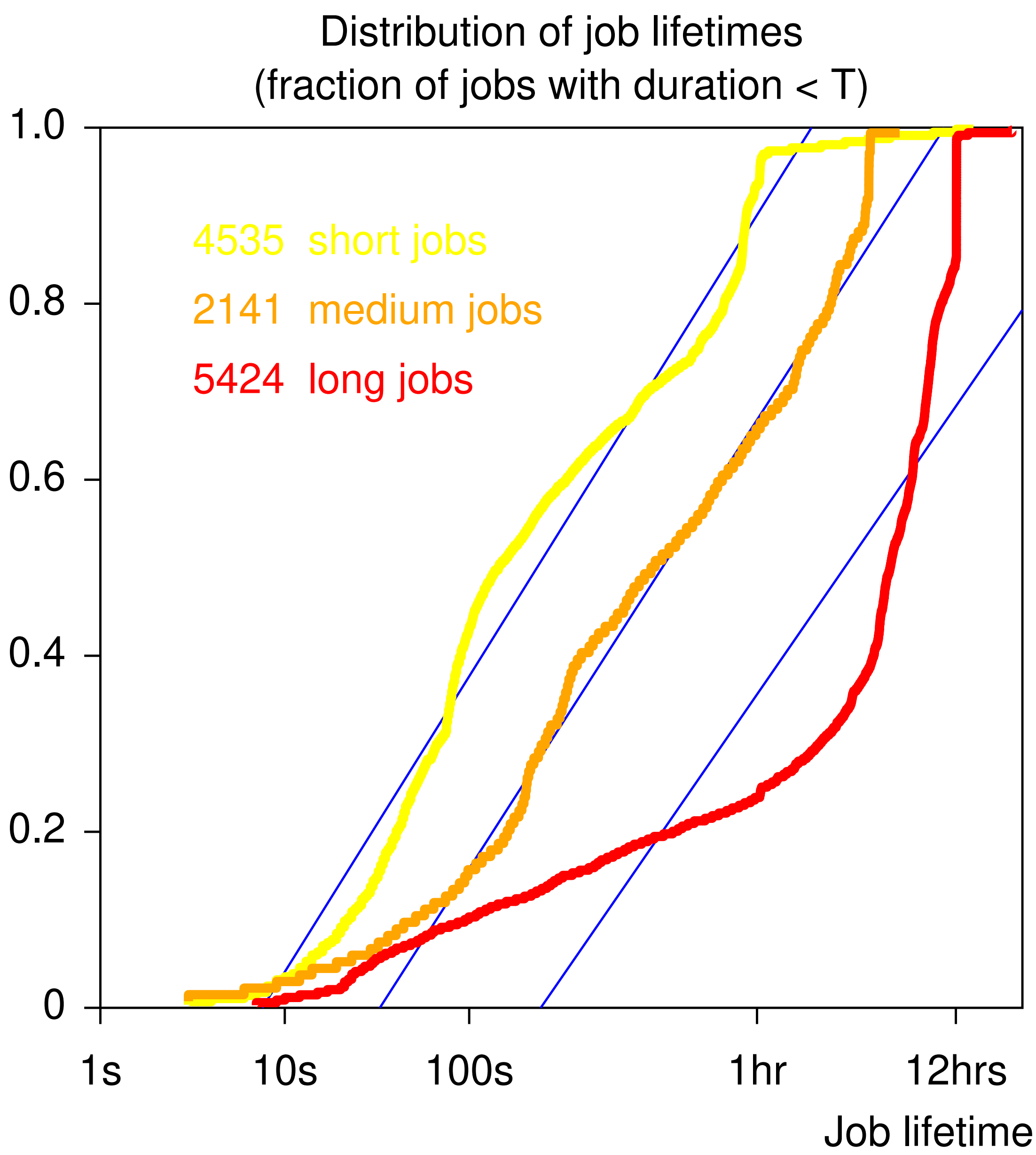
$$\boxed{Prob\ \{0x0\} \approx 1 - P(t)}$$

$O(p)$ $O(2^p)$

Trace-driven simulation

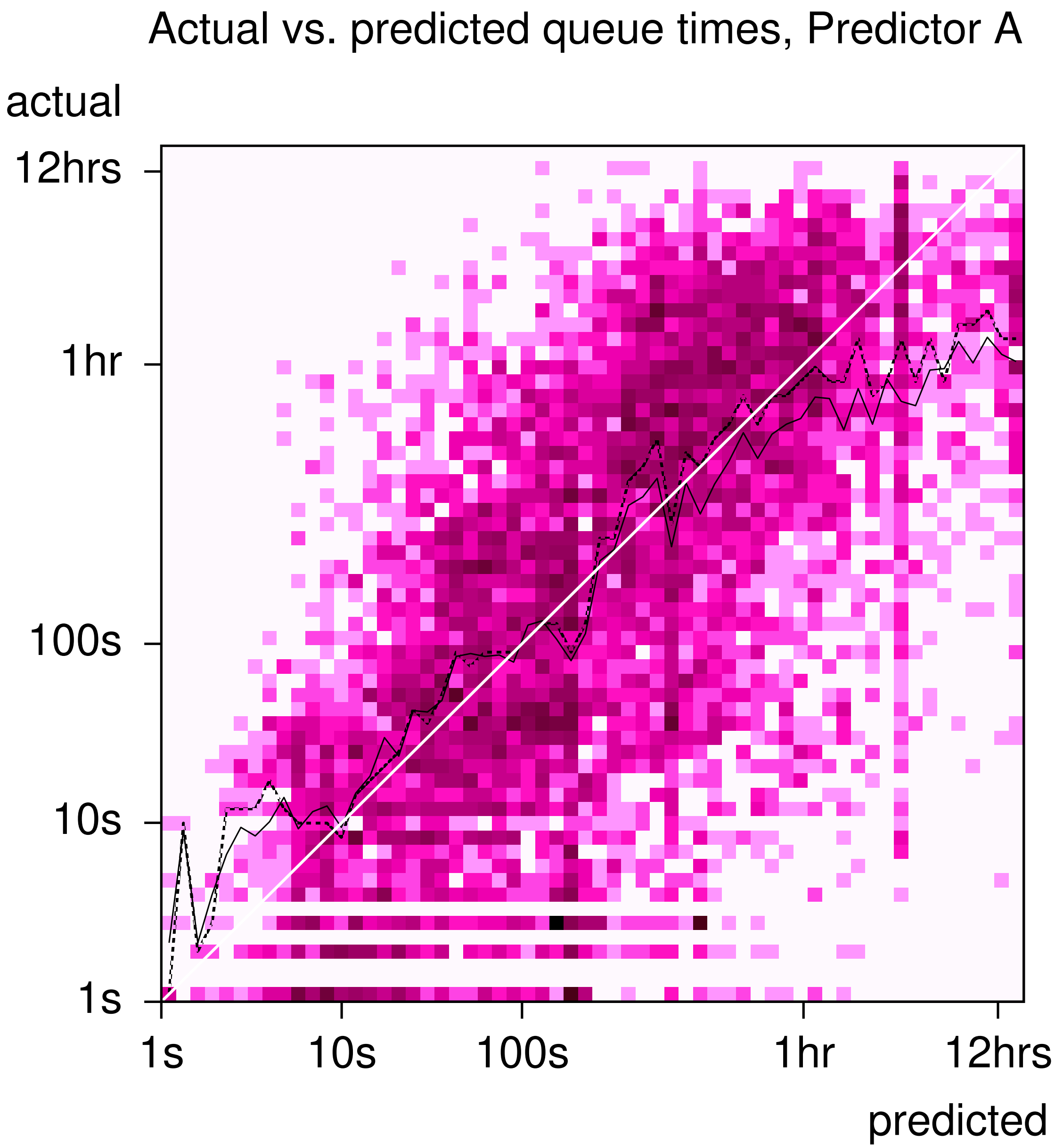


Breakdown by queue name (SDSC)



- Each queue is different, but all are uniform-log.
- Except long.
- Information from users not all that reliable.

Results of Predictor A (SDSC)

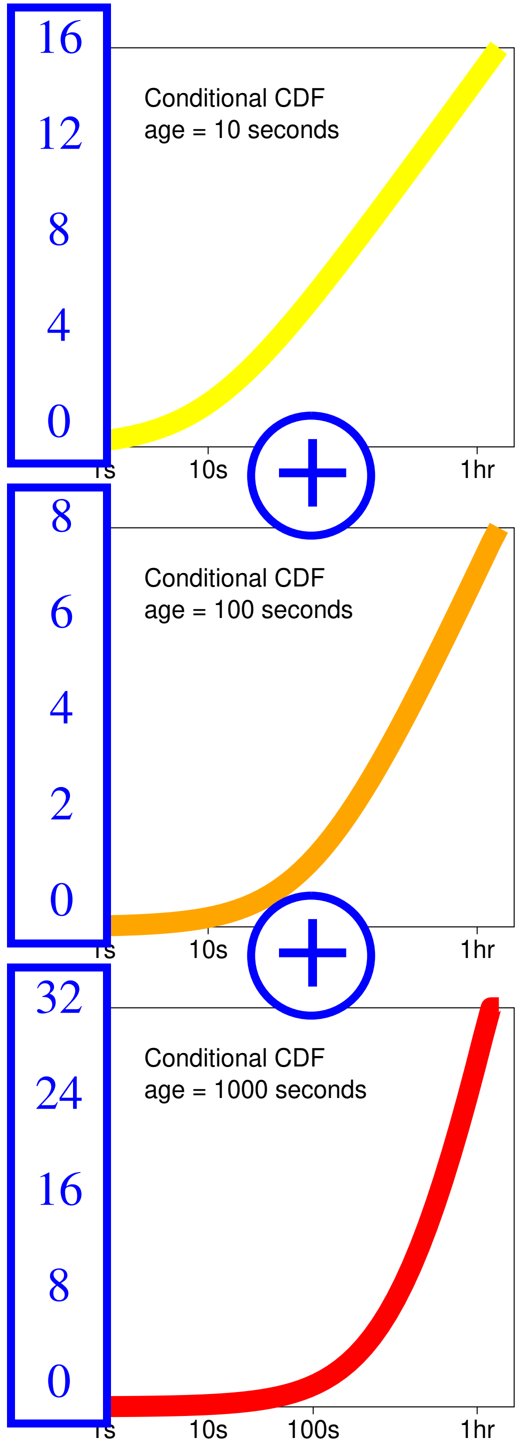


- Coefficient of correlation = 0.63
- Few extreme errors.
- Mean and median of each column near identity.
- Distribution in each column is symmetric.
- For large requests, predicted times too high.

Predictor B

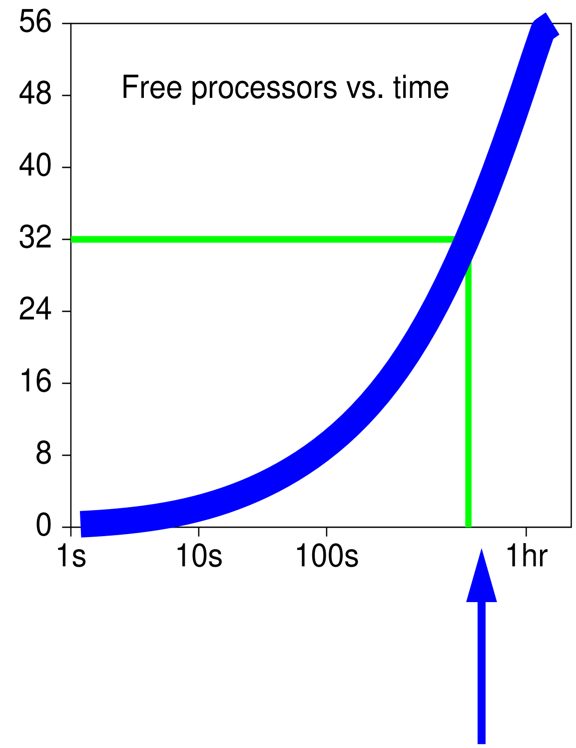
Estimate E[$Q(n)$].

Job	age
1	10
2	100
3	1000



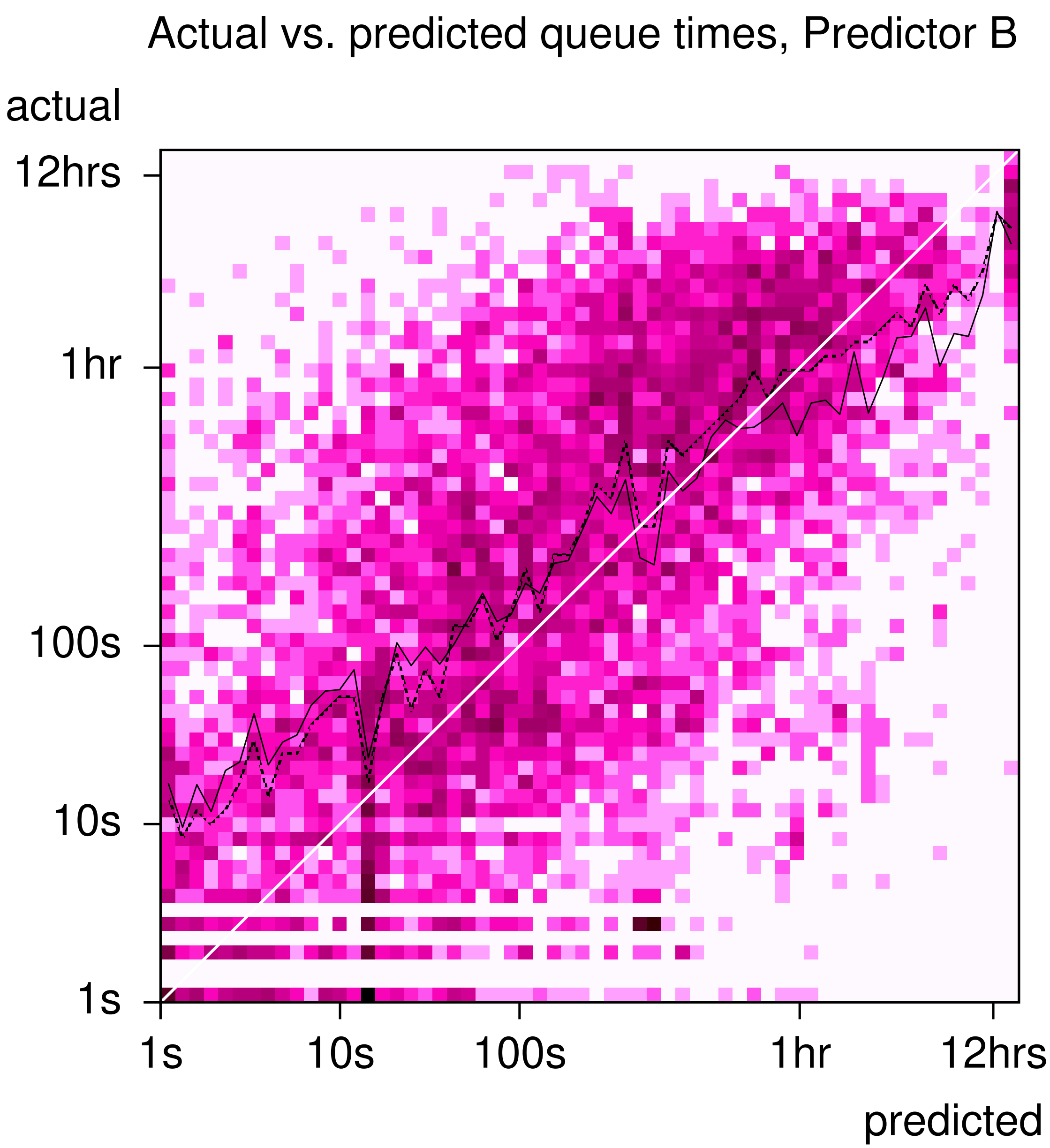
=

Estimate number of free processors versus time...



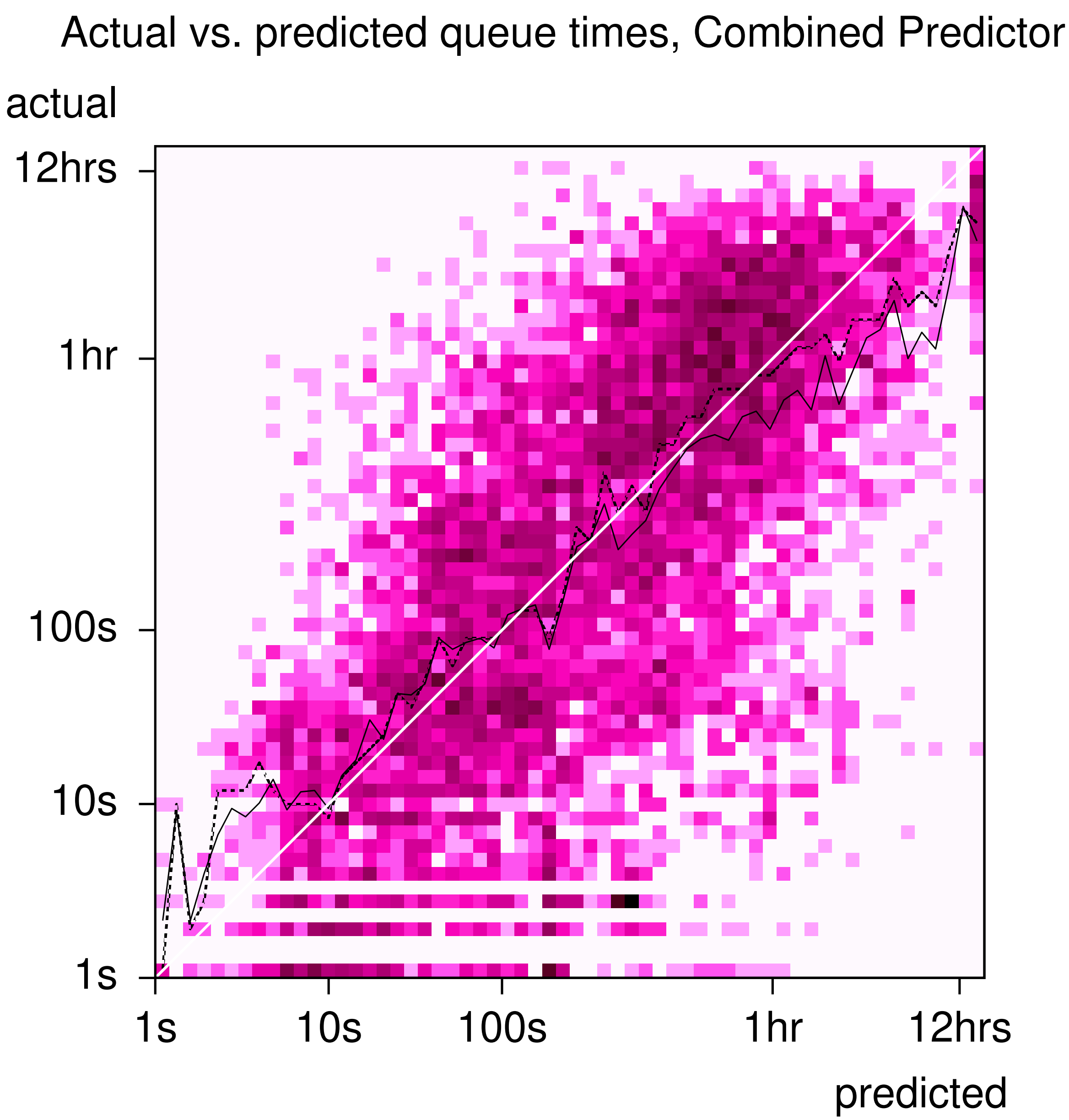
... and invert: 20 min.

Results of Predictor B (SDSC)



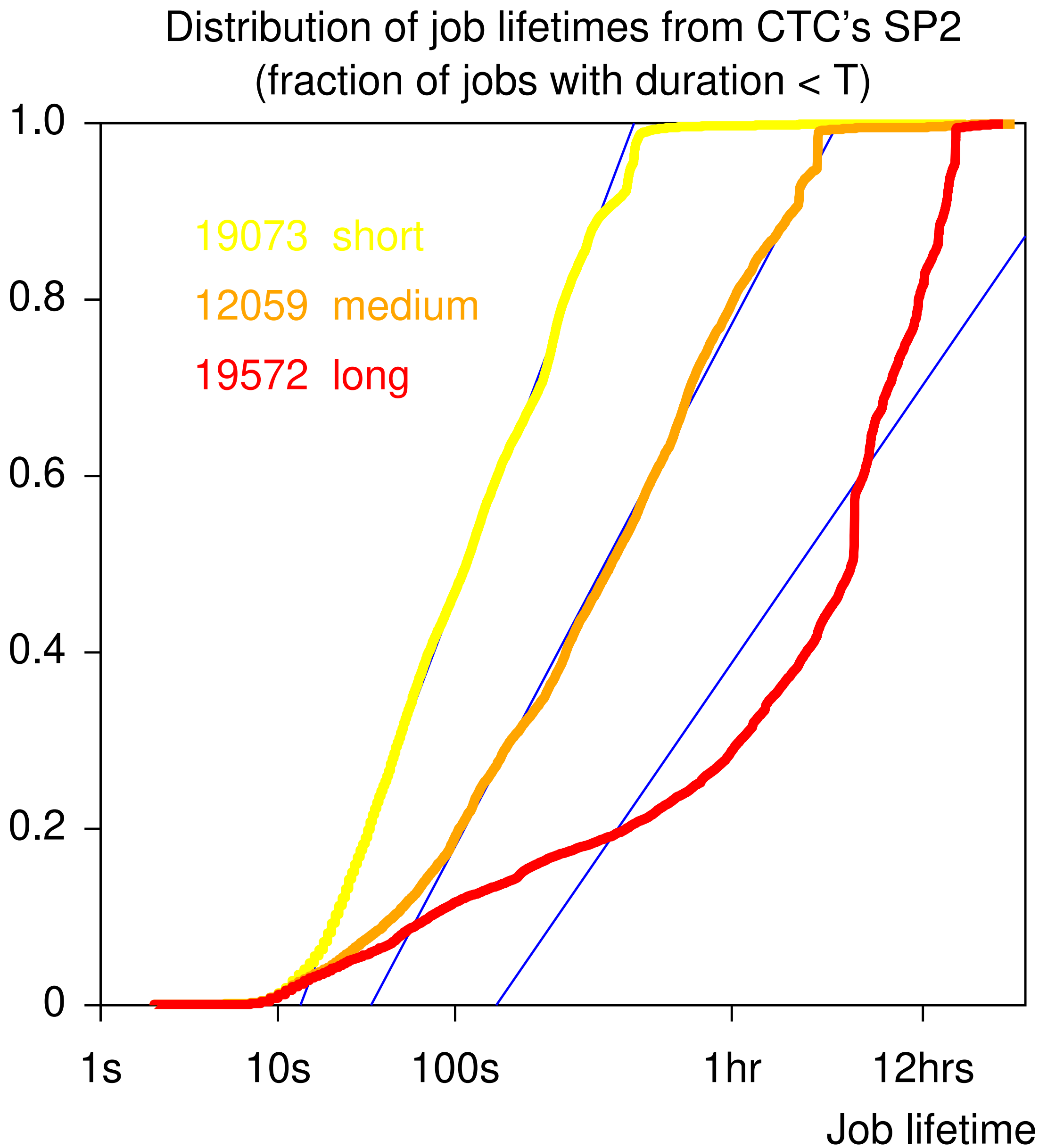
- Coefficient of correlation = 0.61
- Better for small requests
- Still conservative for large requests.

Results of Combined Predictor (SDSC)



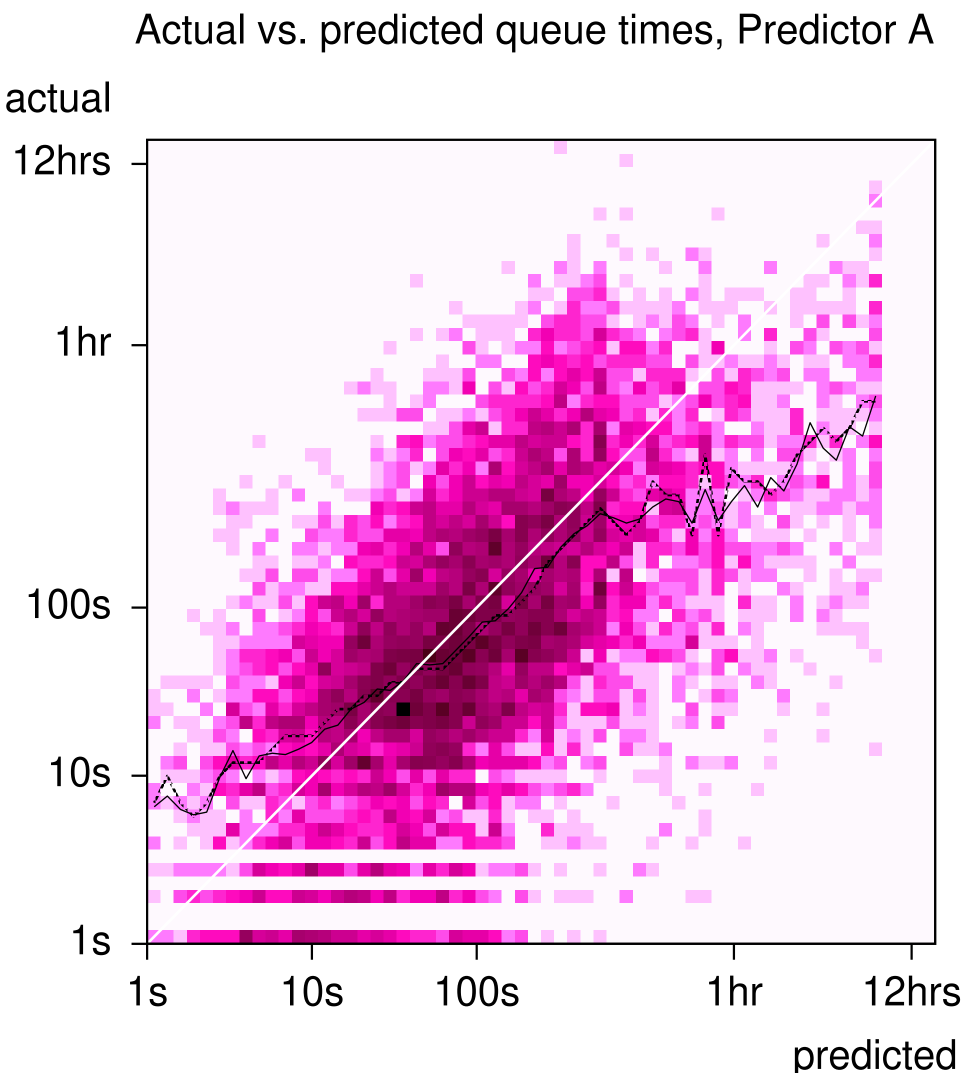
- Predictor A for small requests; Predictor B for large requests.
- Coefficient of correlation = 0.65

Breakdown by queue name (CTC)

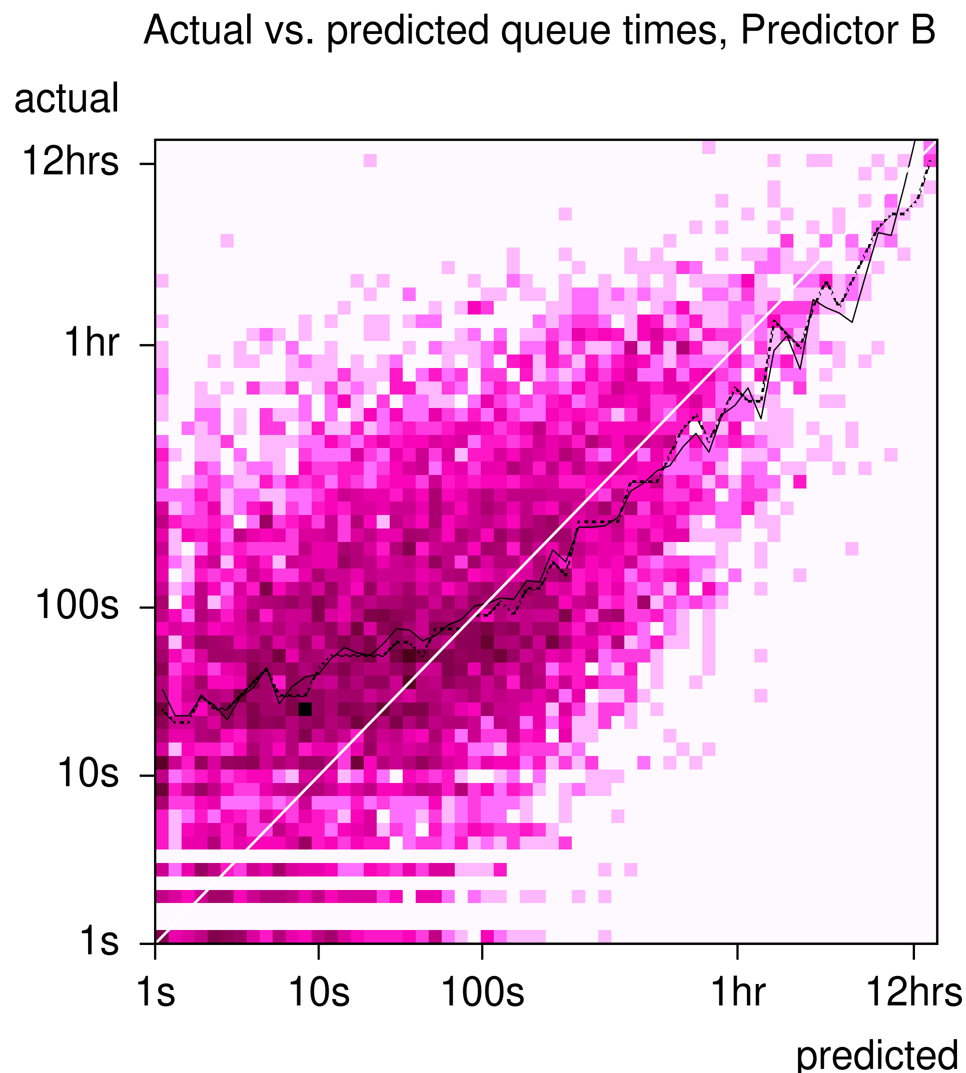


- Less overlap between queues.
- Long distribution less bimodal.
- Better quality information.

Results for IBM SP2 at CTC

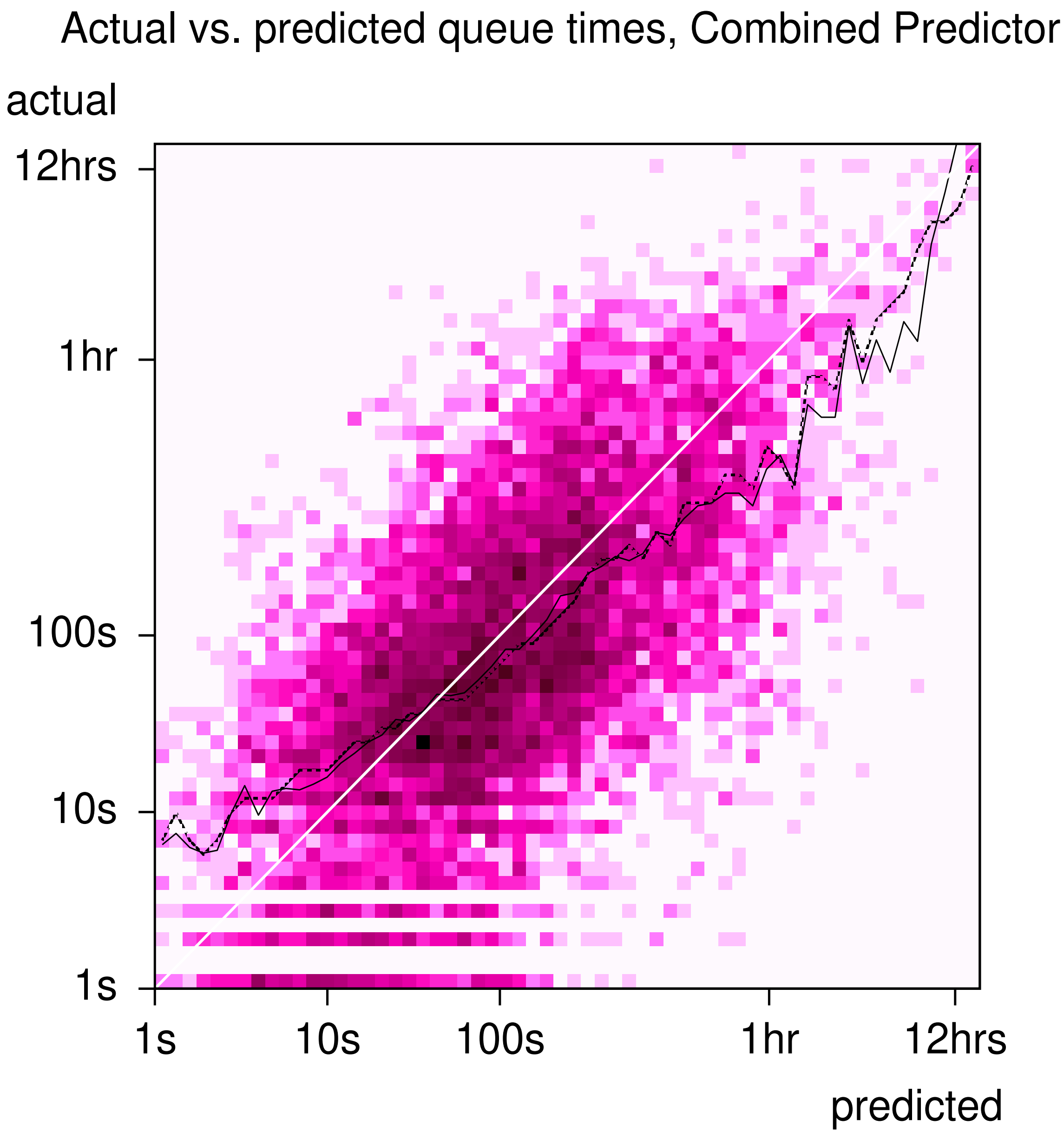


CC = 0.68



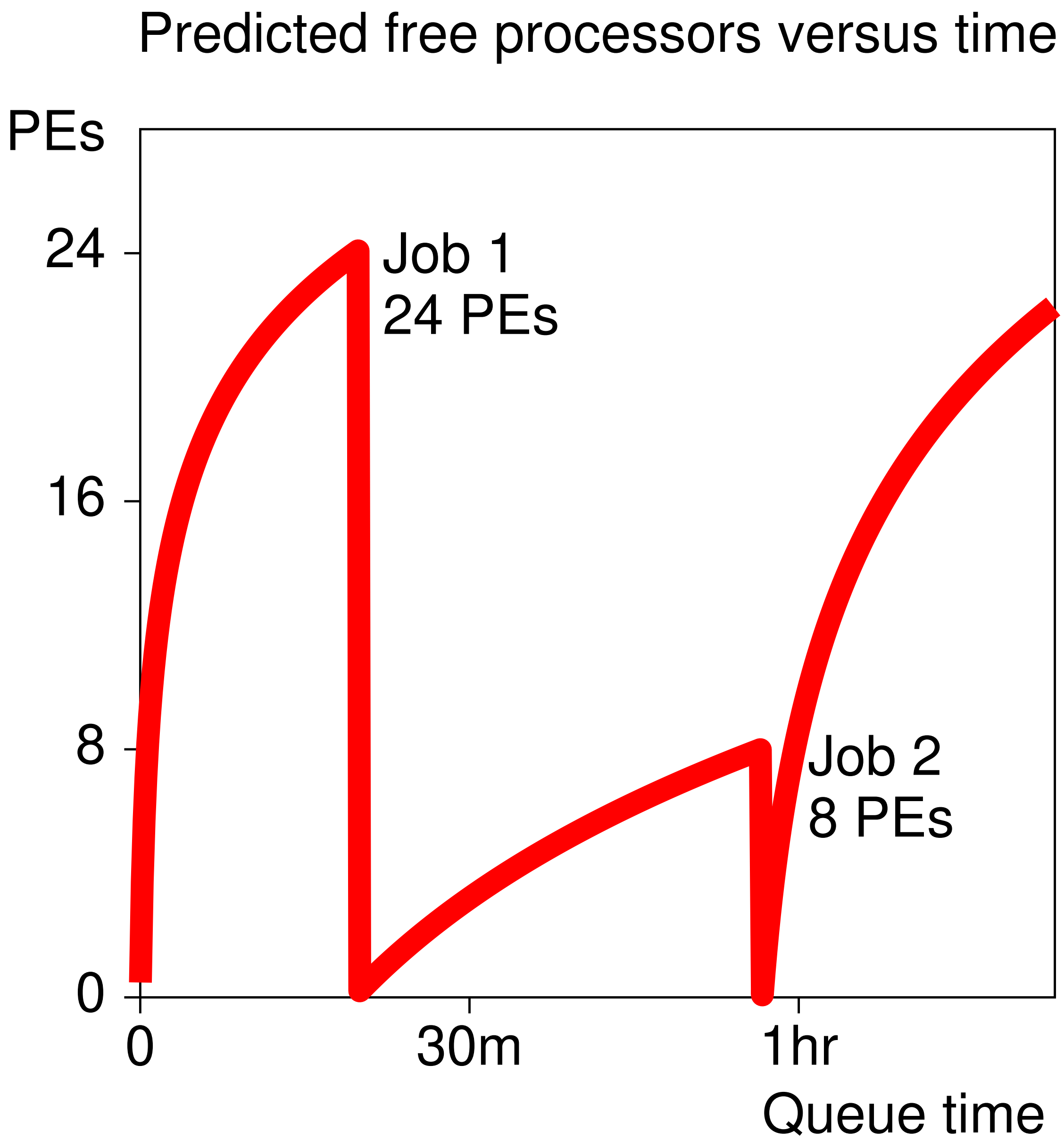
CC = 0.67

Results of Combined Predictor (CTC)



- CC = 0.72
- Again, a little conservative for large requests.

Future work



- Make predictions from end of queue.
- DOCT: Legion object to answer QOS queries.
- Issues — how many classes, how often do we update parameters?