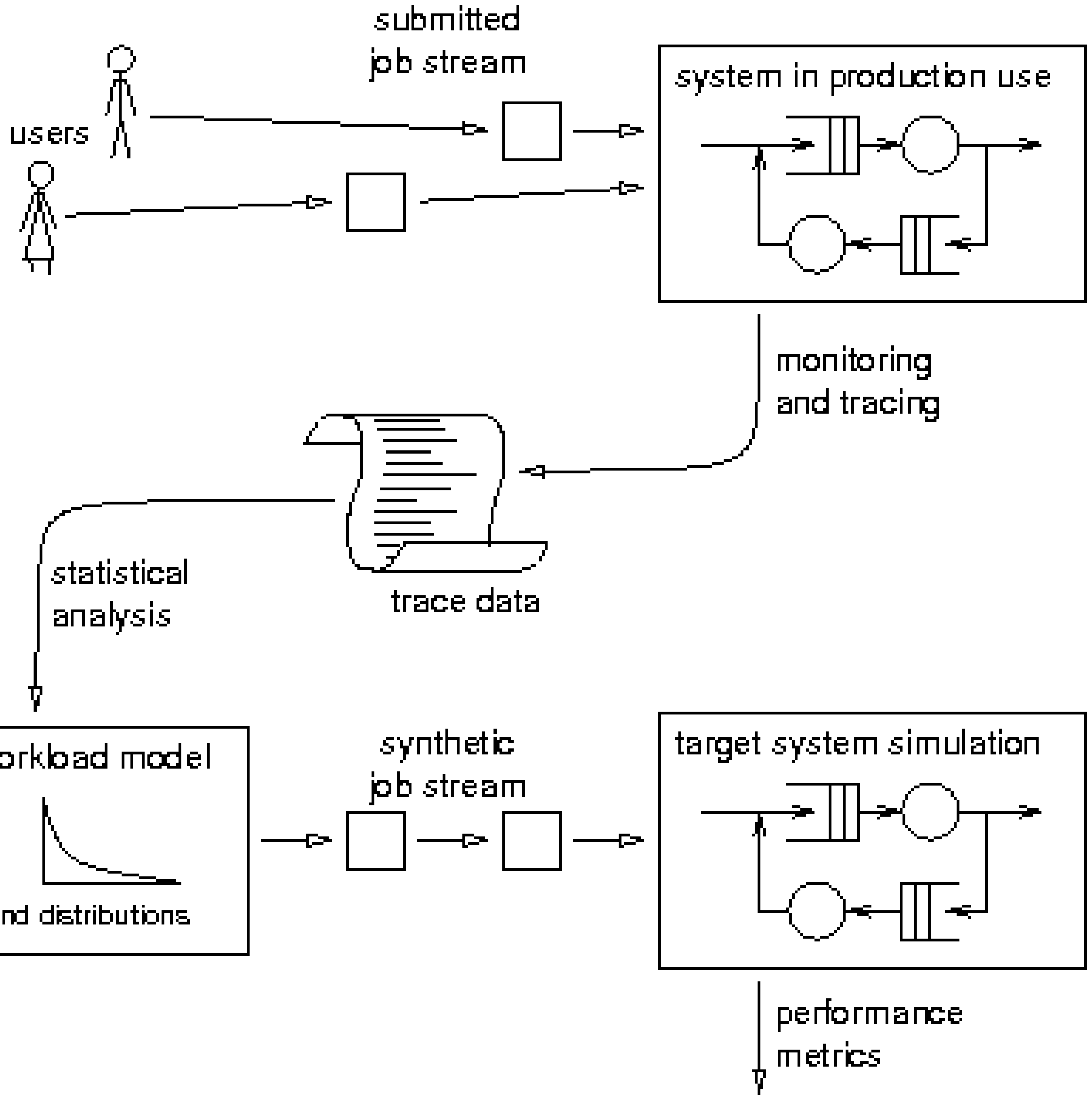
Topics in Performance Evaluation

Feedback and User-Based Workload Models

Dror Feitelson
Hebrew University

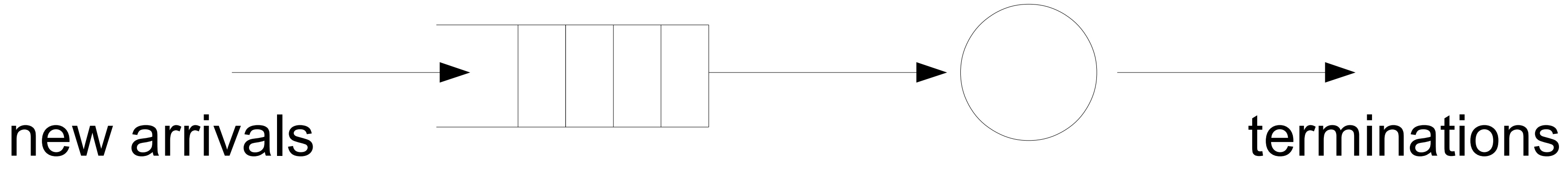
Workload modeling based on trace data:



Typical attributes modeled by distributions:

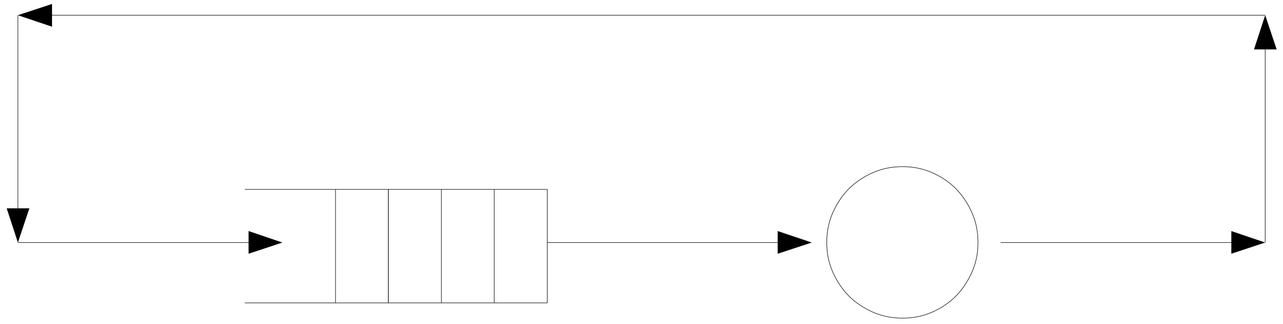
- Job arrival times
- Job runtimes
- Job sizes (resource requirements)

Model of arrival times is open:
jobs arrive from an infinite population of users, independent of system state



Also applies to re-playing a job trace

Alternative is a closed model:
jobs only arrive when a previous one terminates, so completely dependent on system state



Neither model is very realistic:

- open model ignores effect of system on new arrivals – they continue to arrive even if the system is overloaded
- closed model assumes the number of jobs is always constant

A more realistic approach: generative modeling

- Do not model the workload as it is observed
- Instead, model the process that generates the workload
- Process is not oblivious, and may include interactions with the system

Leads to site-level simulations:

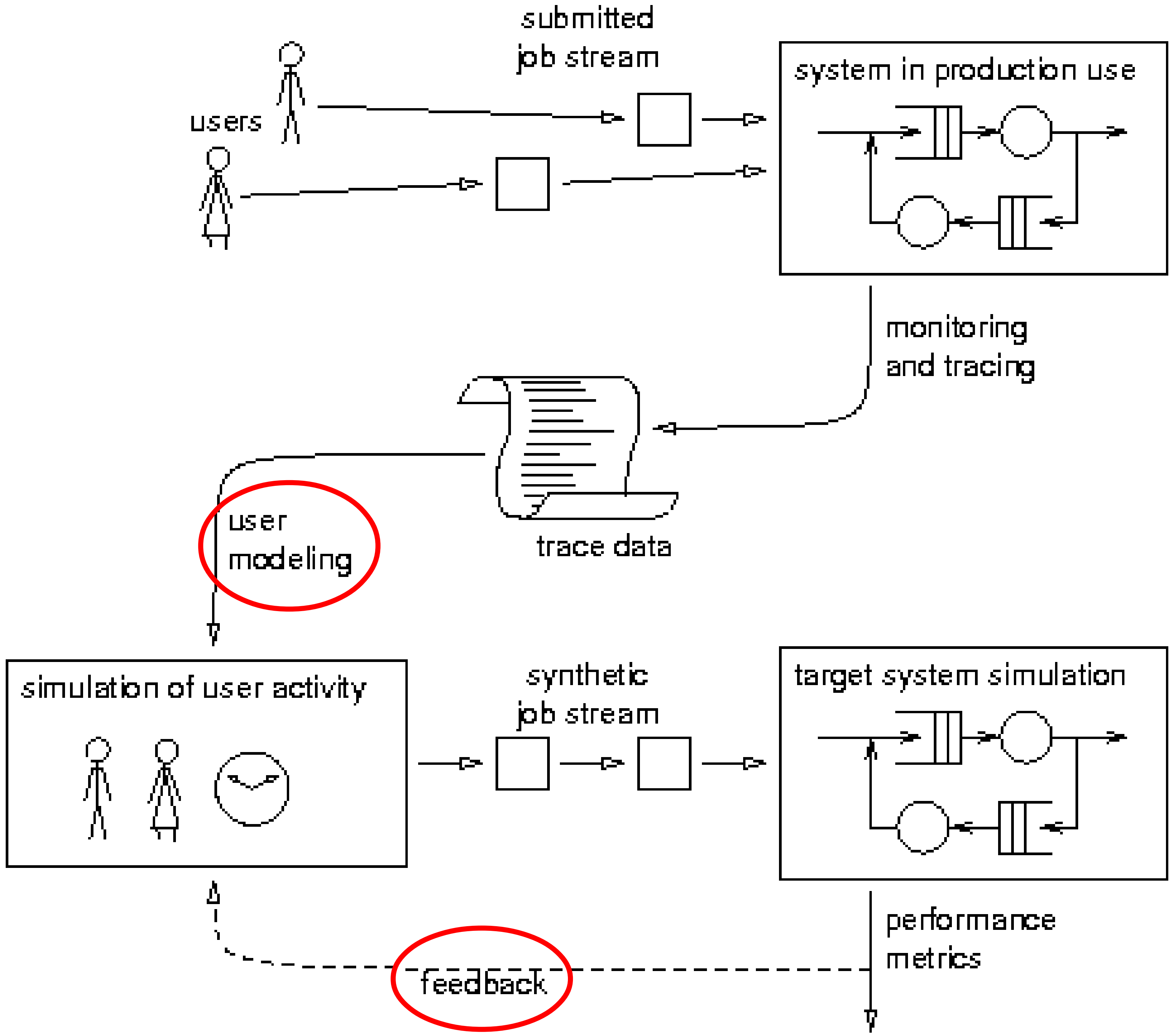
to evaluate a system, we also need to simulate its environment (=its users)

Specific example: user-based modeling

- Model the behavior of users as they generate the workload
- Including how users adapt their behavior in response to system performance
- This requires feedback from the system
- Also model the dynamics of the user population

Another example: network traffic modeling

- Model behavior of TCP as it sends packets over the network
- Includes sliding window and congestion control
- Called “modeling at the source”


submitted job stream
users
system in production use
monitoring and tracing
trace data
user modeling
simulation of user activity
synthetic job stream
target system simulation
performance metrics
feedback

Generative model is often hierarchical

- Web usage
  - User session (arrivals, length)
  - Requests for pages (sequence, links)
  - Requests for embedded objects (bursts)
- Database usage
  - Enterprise-level application logic
  - High level queries
  - Bursts of I/O operations
- User population, sessions, and activity [elaborated below]

Site-Level Simulation

Shmueli and Feitelson, Using site-level modeling to evaluate the performance of parallel system schedulers. 14th MASCOTS, Sep 2006

Conjecture:

- If the system performs well, users will submit more jobs
- If the system performance is lousy, users will refrain from submitting additional jobs

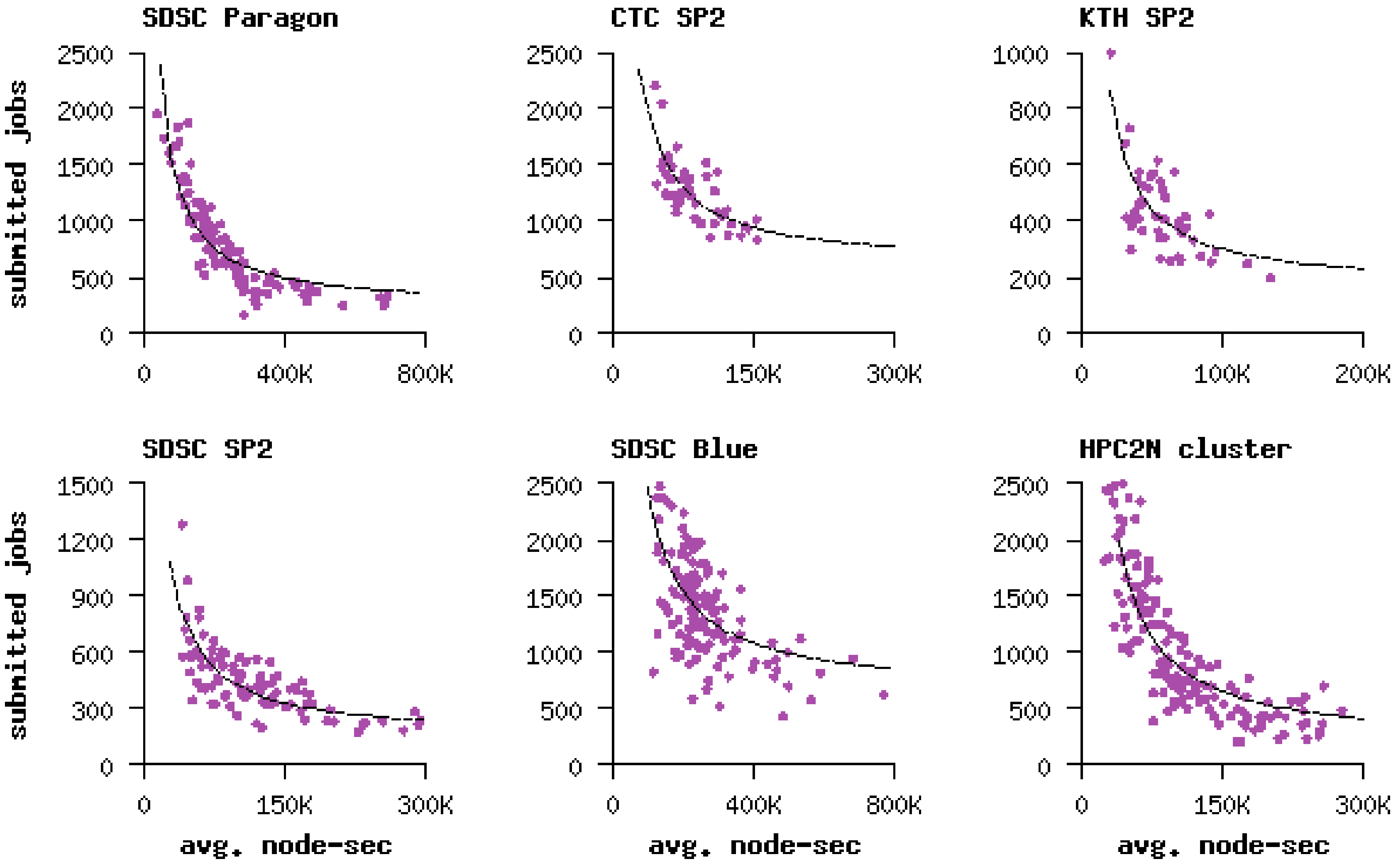
So systems are expected to display self-throttling:

submission rate is dictated by system load and its ability to cope with that load

and not by predefined timestamps

[Following examples from parallel job scheduling]

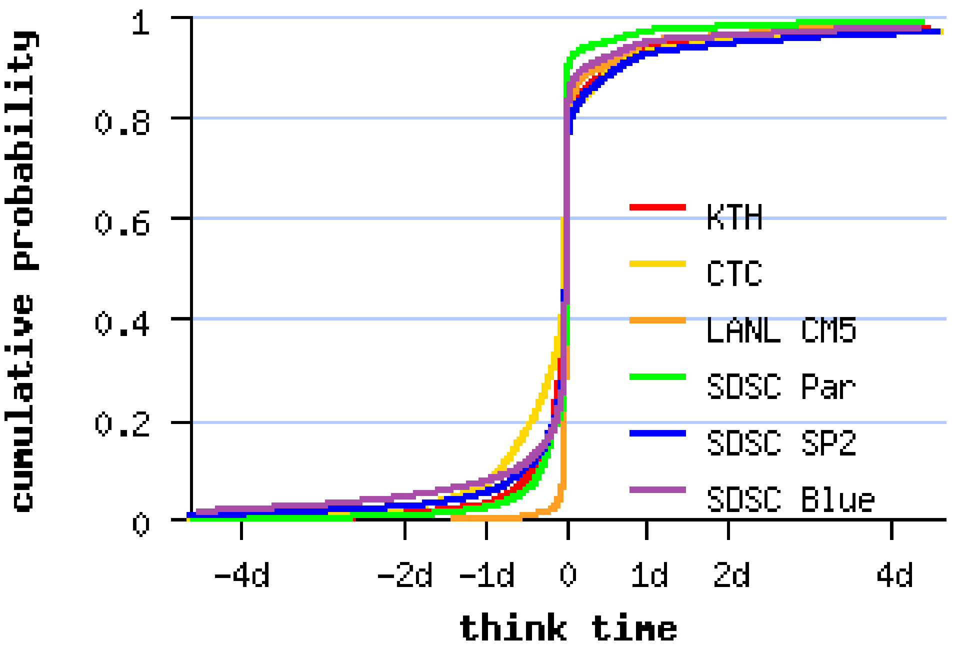
Evidence:

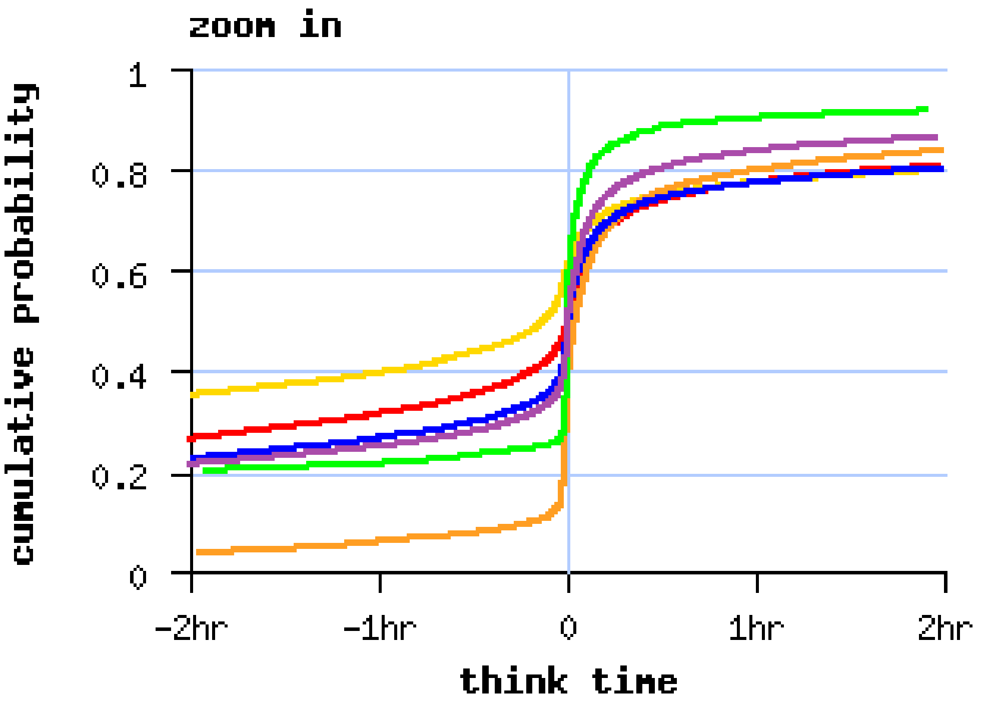

SDSC Paragon
CTC SP2
KTH SP2
SDSC SP2
SDSC Blue
HPC2N cluster
submitted jobs
avg. node-sec
0
400K
800K
150K
300K
100K
200K
500
1000
1500
2000
2500
200
400
600
800
300
900
1200

To introduce feedback, need model of user behavior in an interactive session

Learn about this from traces of real systems

In particular, look at distribution of think times (from termination of one job to arrival of the next one)





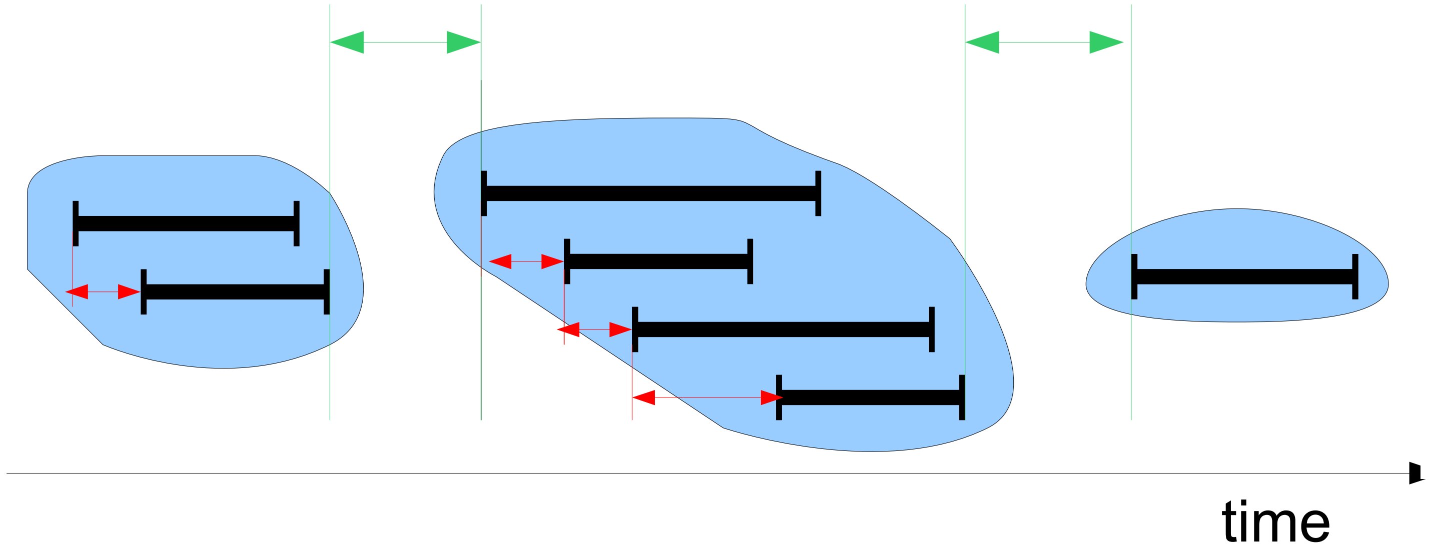
Surprise: many are negative

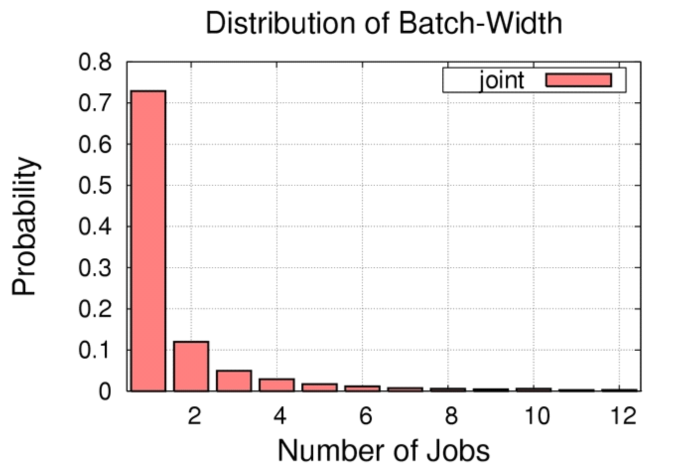
Conclusion: users submit jobs in batches

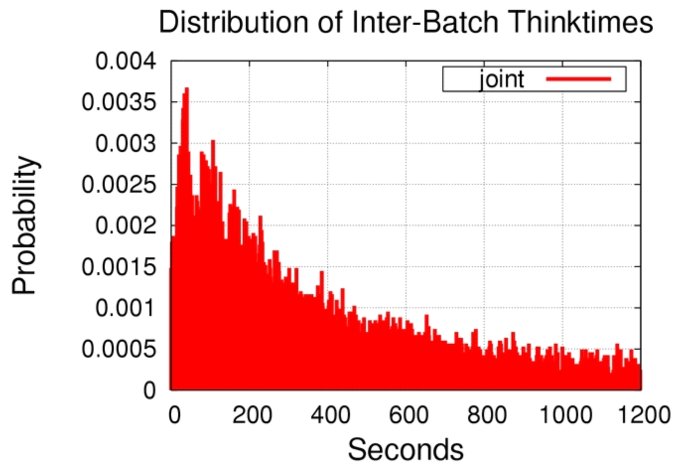
Need data about batch width

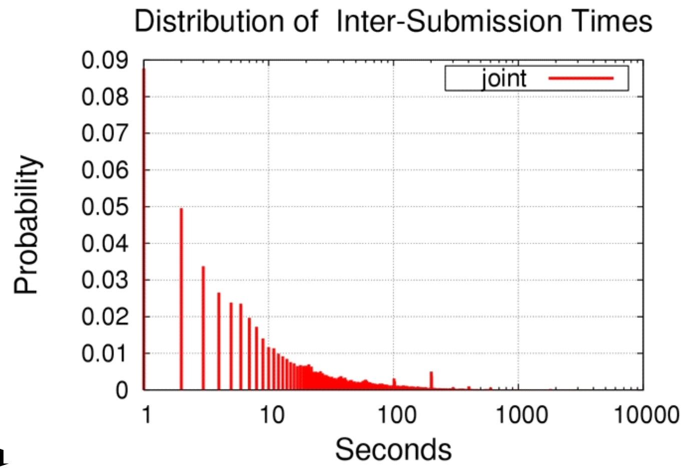
Think times between batches (this is where feedback comes)

Inter-arrivals within each batch

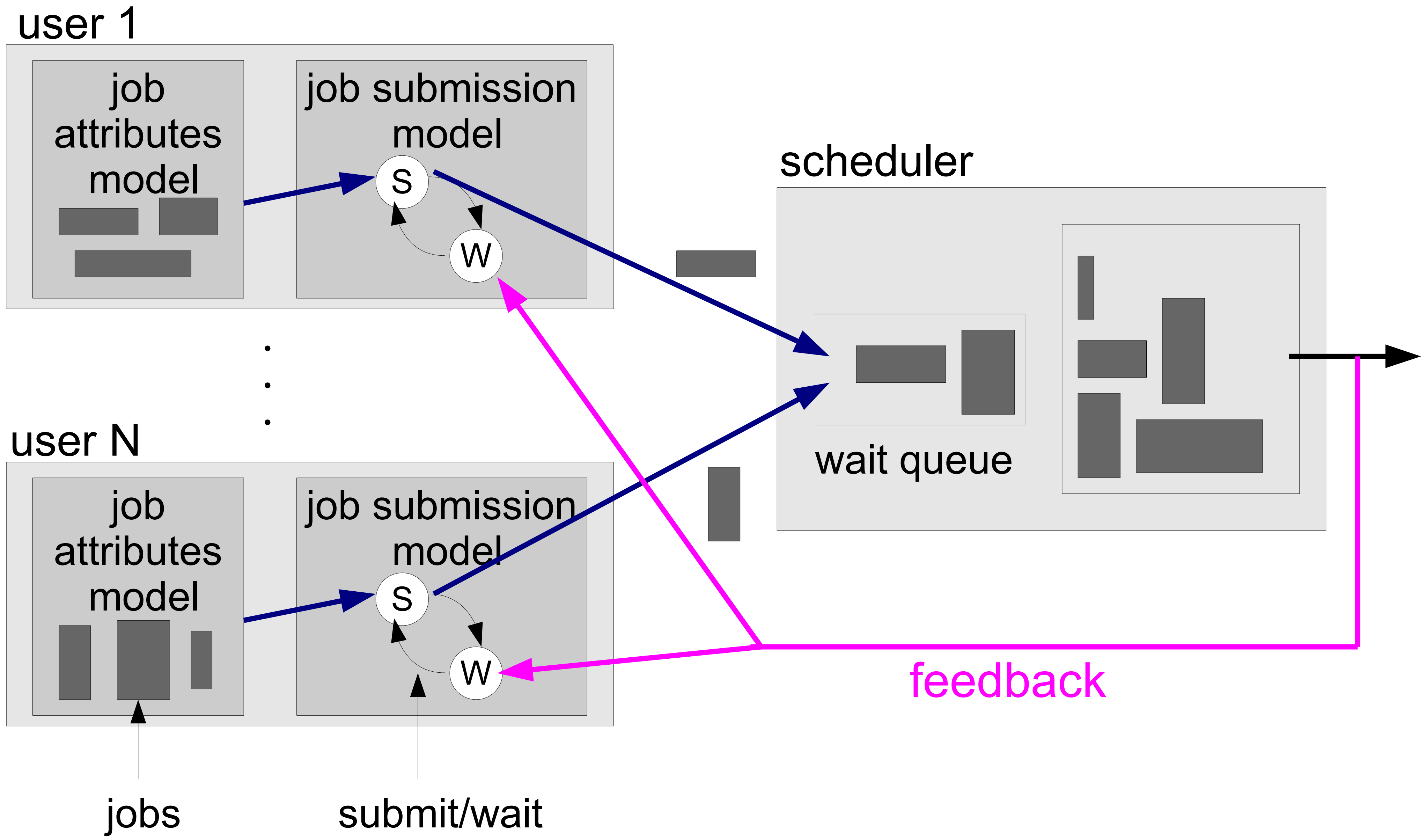








Complete site-level simulation:



Importance of Feedback

Shmueli and Feitelson, Using site-level modeling to evaluate the performance of parallel system schedulers. 14th MASCOTS, Sep 2006

Conjecture: a workload trace reflects

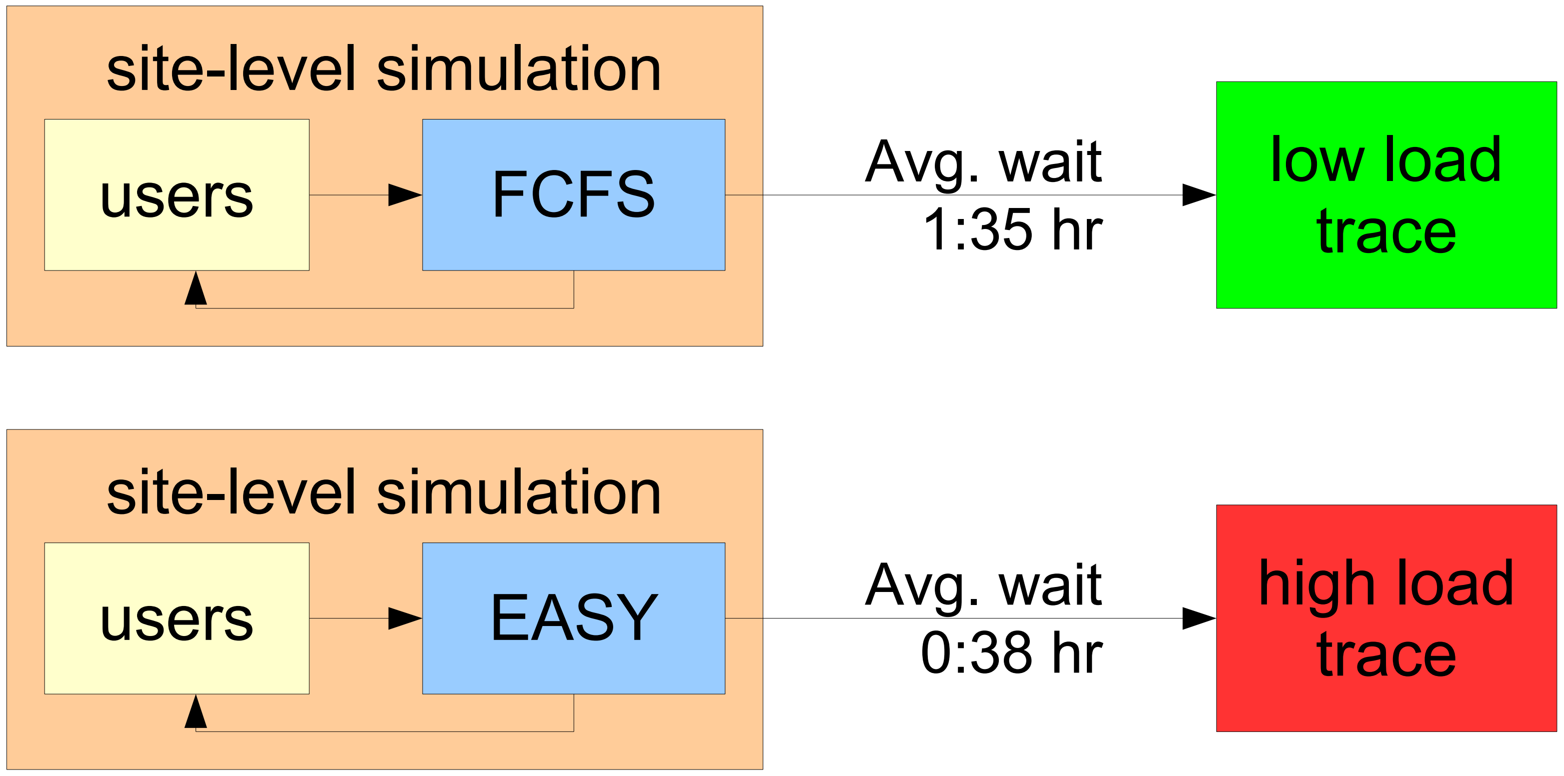
1) The behavior of the scheduler that was used on the traced system
2) The interaction between the scheduler and the users

So using such a trace to simulate another scheduler leads to unreliable results!

We'll demonstrate this using traces generated using site-level simulation

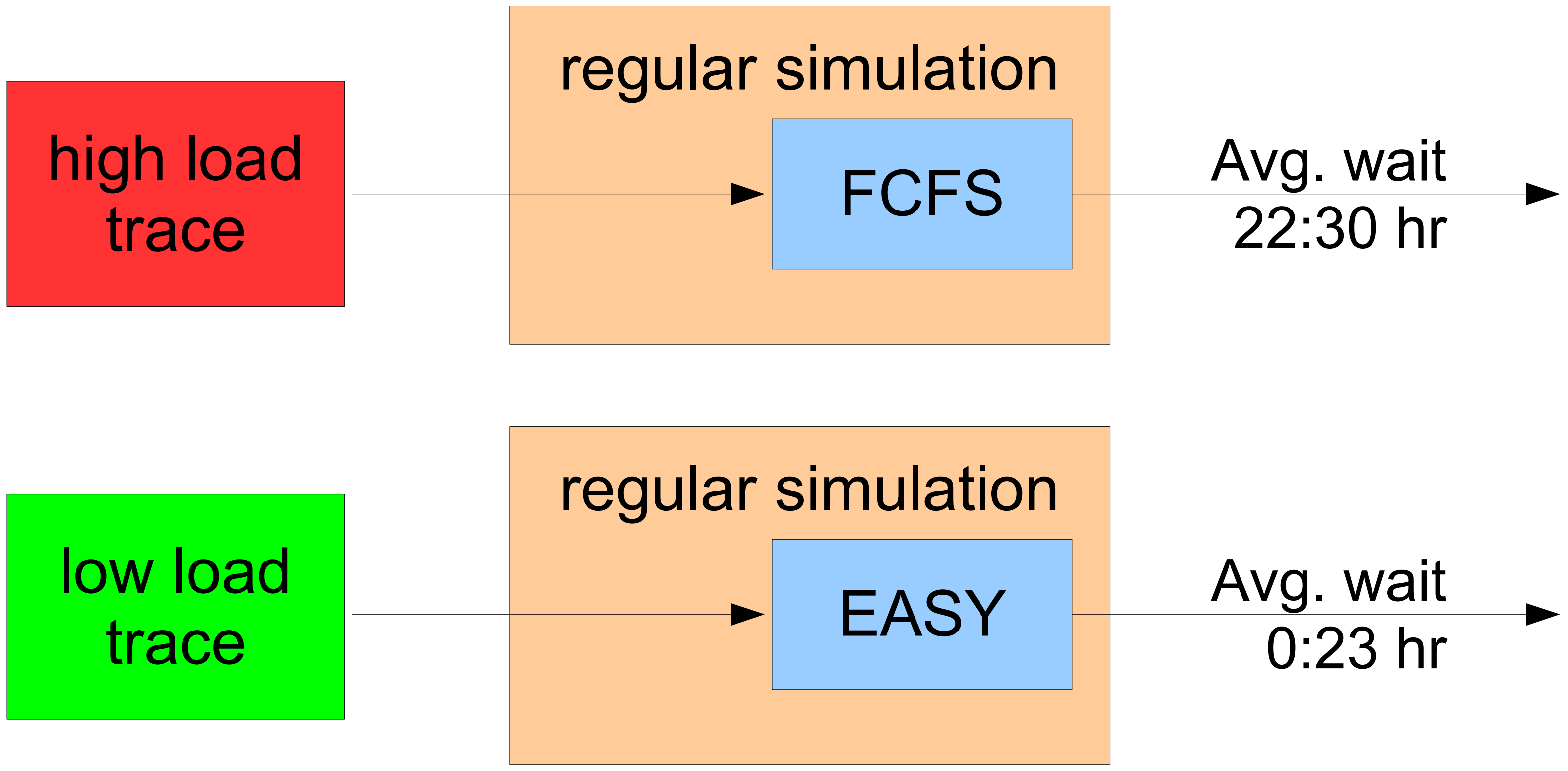
Step 1: generate traces

FCFS – a simple scheduler that cannot support a high load
EASY – a more efficient backfilling scheduler



Step 2: switch traces and use in regular simulation

simulate FCFS with high-load trace generated using EASY
simulate EASY with a low-load trace generated using FCFS



Comparison of results:

Scheduler	*Site-level*	*Wrong trace*	*Difference*
FCFS	01:35	22:30	+1345%
EASY	00:38	00:23	-38%

So results of regular simulations based on a trace from another scheduler are unreliable
(assuming you accept that site-level simulations are reliable)

Additional problem: violation of dependencies between jobs

Assuming the batch submissions model, the jobs in each batch depend on those in the previous batch

But in trace-based simulations these dependencies may be violated: a job's arrival timestamp is before termination of previous job

May even arrive before previous job was started!

This happens more often when the load is increased by reducing interarrival times

User-Based Workload Model

Typical workload models characterize the jobs directly. This leads to several problems:

1) Not parametric (needed for tuning to specific conditions)
2) No good way to manipulate load
3) Do not provide locality of sampling (regularity and predictability)
4) No daily/weekly cycle
5) No self similarity (burstiness at different time scales)
6) No support for feedback
7) Do not support unique behavior and anomalies (flurries)

The alternative: a user-based generative model

Three level user-based generative model:

1) User population model
   How new users arrive and depart the system

2) User sessions model
   The activity patterns of a single user, including the daily cycle

3) User activity model
   How users submit jobs within a session
   [this is the part we did before]

Combines open and closed system models

Goals of the user population model:

User arrivals and departures cause fluctuations in load – not always the same number of active users

User arrivals and departures cause changes in workload patterns, as different users have different personalities

Provide some control over load (more users →higher load)

The overall mix should reflect variability seen in real systems

Specifics of the user population model:

Match data regarding new user arrivals

Match data regarding user residence times (implies user departure times)

Match data regarding mix of user personalities (activity on workdays or nights/weekends, level of activity, etc.)

Allow for unique/abnormal users (e.g. flurries, bots)

Goals of user sessions model:

Create realistic fluctuations in load
(heavy-tailed session times/breaks lead to self similarity)

Align activity of different users
(all relate to the same daily cycle)

Incorporate some feedback
(aborted sessions due to bad performance)

Specifics of user sessions model:

Session starts should match data, and may be different for different user personalities

A session can end for one of 3 reasons:

1) It is the end of the work day and time to go home
2) The session has been going on for a long time and the user is tired
3) The system's performance is not good and the user gave up
   [note that this implies feedback]

Goals of user activity model:

Create (short term) realistic fluctuations in load

Create locality due to repeated actions of users (important for predictability and evaluation of adaptive systems)

Incorporate main feedback effects

Specifics of user activity model: locality of sampling

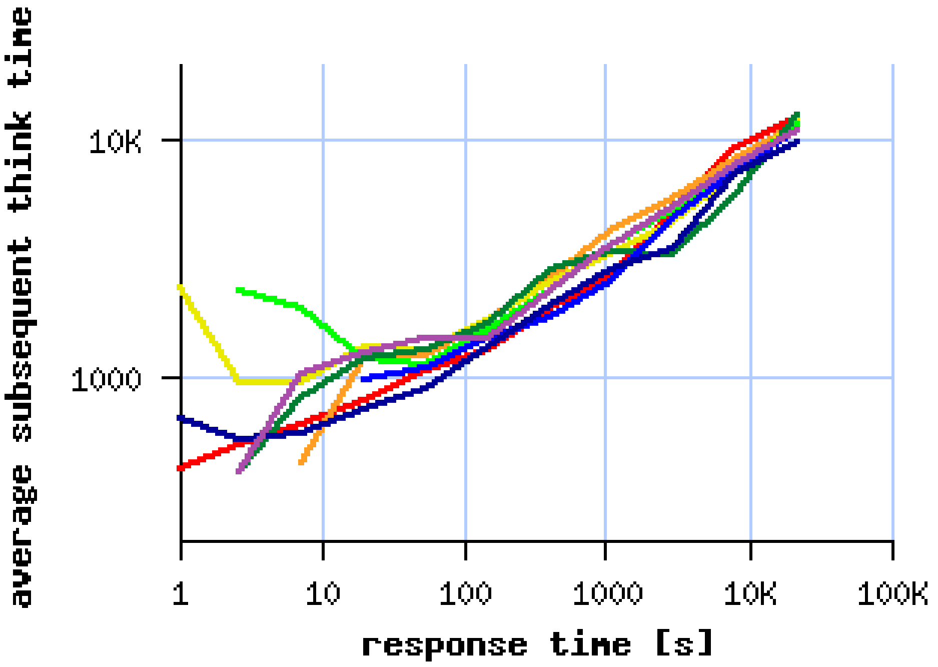
Users tend to repeat the same work over and over again

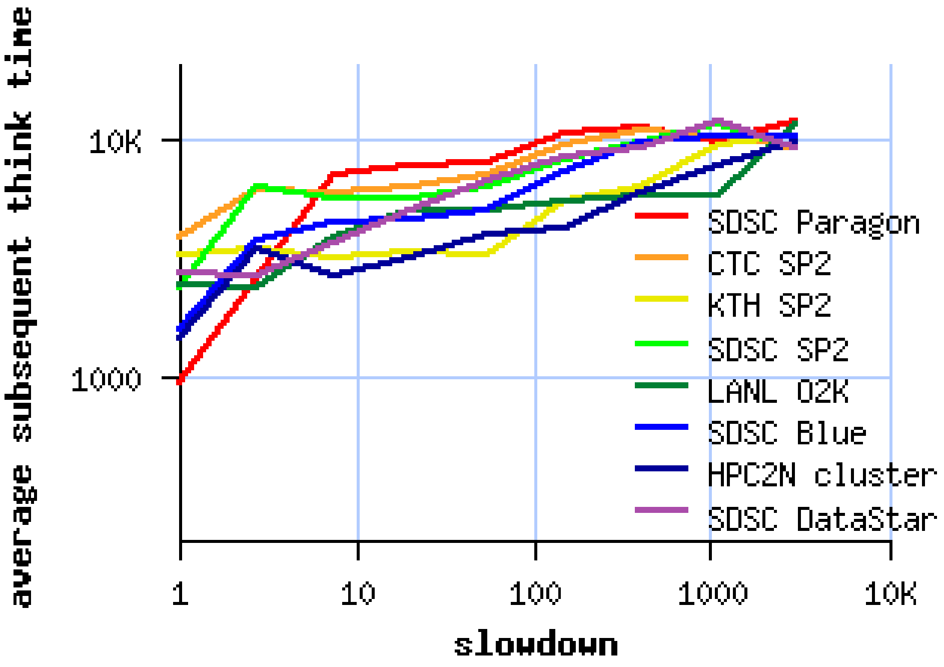
Therefore workloads are actually much more predictable than we may think

This is important for adaptive systems that learn about their environment and optimize their behavior for it

So workload models must include repetitions too

Specifics of user activity model:
Realistic user think times: what do users care about?





Average think times grow with response time!

An Alternative: Workload Resampling

N. Zakay and D. G. Feitelson, Workload resampling for performance
Evaluation of parallel job schedulers. 4th ICPE, Apr 2013.

Resampling vs. Modeling

- Modeling is hard
  - Multiple models (population, sessions, activity)
  - Multiple distributions and parameters
  - Important things may be left out
- Alternative is resampling
  - Break log into small pieces
  - Resample from pool of pieces to create new workload
  - Can do this many times with variations

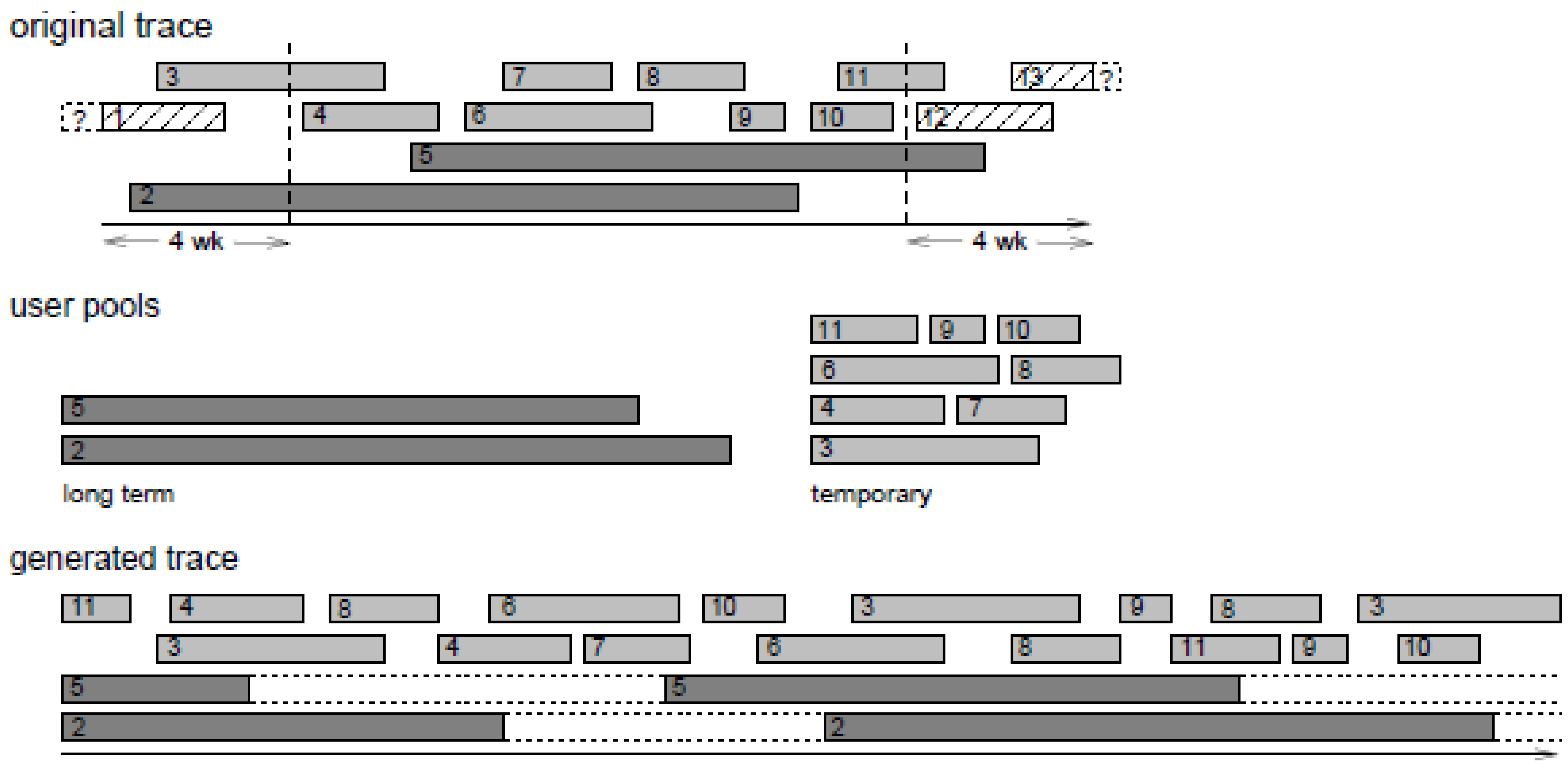
Granularity of Resampling

- Individual jobs
  - Retain correlations of job attributes
  - Mix job order, lose locality
  - Need to model arrivals, sessions
- User sessions
  - Retain locality in sessions
  - Need to model activity patterns
- Complete user activity
  - Retain all locality and patterns
  - Need to adjust arrivals to allow for feedback

We chose User-level resampling

User Classification

- Long-term users
  - Continuously active throughout the log
  - Should be continuously active in simulations
- Temporary users
  - Only appear for a limited time
  - Can be resamples several times in simulation
- Special users (e.g. flurries or bots)
  - Can be removed to simulate normal users only
  - Can be amplified to see their effect

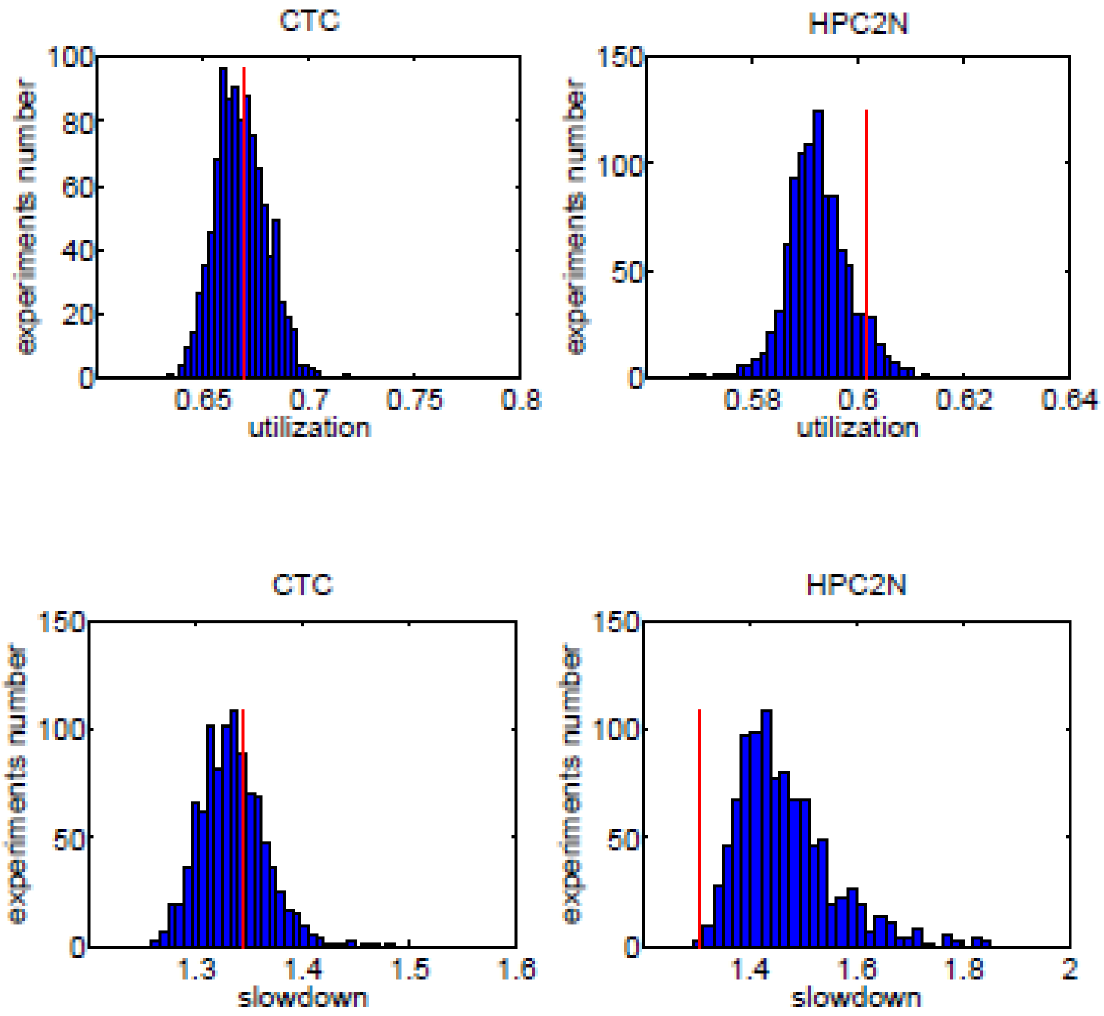

original trace
3
7
8
11
13
?
?
1
4
6
9
10
12
5
2
4 wk
4 wk
user pools
11
9
10
6
8
4
7
3
5
2
long term
temporary
generated trace
11
4
8
6
10
3
9
8
3
3
4
7
6
8
11
9
10
5
5
2
2

Applications

- Extend log by continued resampling to enable longer simulation
- Perform multiple similar repetitions and compute confidence intervals on metrics
- Change load by having more or less users
- Combine data from multiple logs for better representativeness
- Remove or emphasize special users

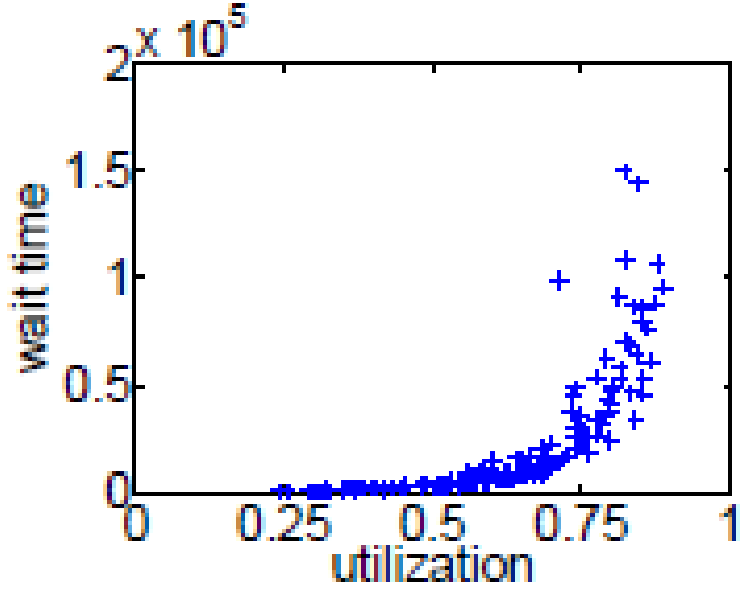
Repeated Simulations

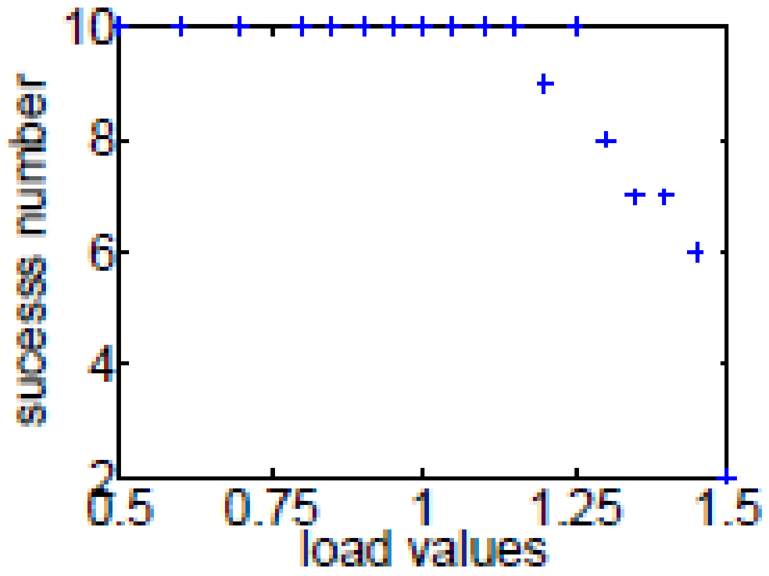
- Verification: resulting load should be similar
  - Sometimes it tends to be lower or higher
- Results: use distribution of performance metrics
  - Simulation with original log may not be very representative



Increased Load

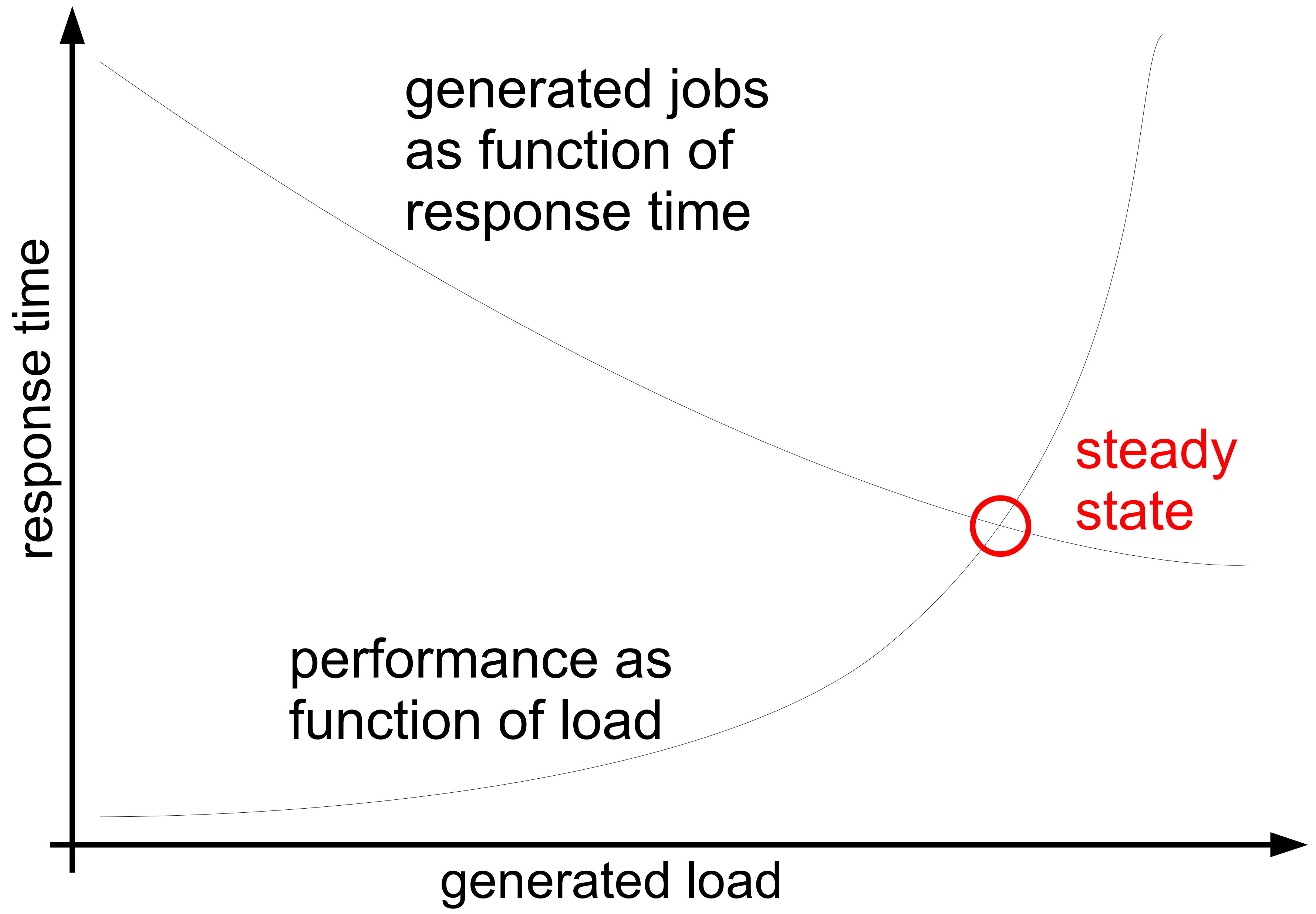
- Individual runs will have slightly different loads
- At high loads simulation may be unstable
  - Detect unstable simulations by increasing queue length
  - Discard results





Feedback and Evaluations

Feedback effects determine the load that will be placed on a system:



Feedback effects on the use of performance metrics:

metric	*oblivious*	*with feedback*
load	should match offered load	metric of performance
throughput	dictated by arrivals	metric of performance
response time	metric of performance	misleading without noting throughput
user satisfaction	represented by response time	represented by throughput

Conservative Modeling and Daily Cycles

Feitelson and Shmueli, A case for conservative workload modeling: parallel job scheduling with daily cycles of activity.17th MASCOTS, Sep 2009

A workload model should include all important workload features

What is important?

- Everything that you know of
- Everything that actually affects the evaluation

The best practical approach is to be conservative

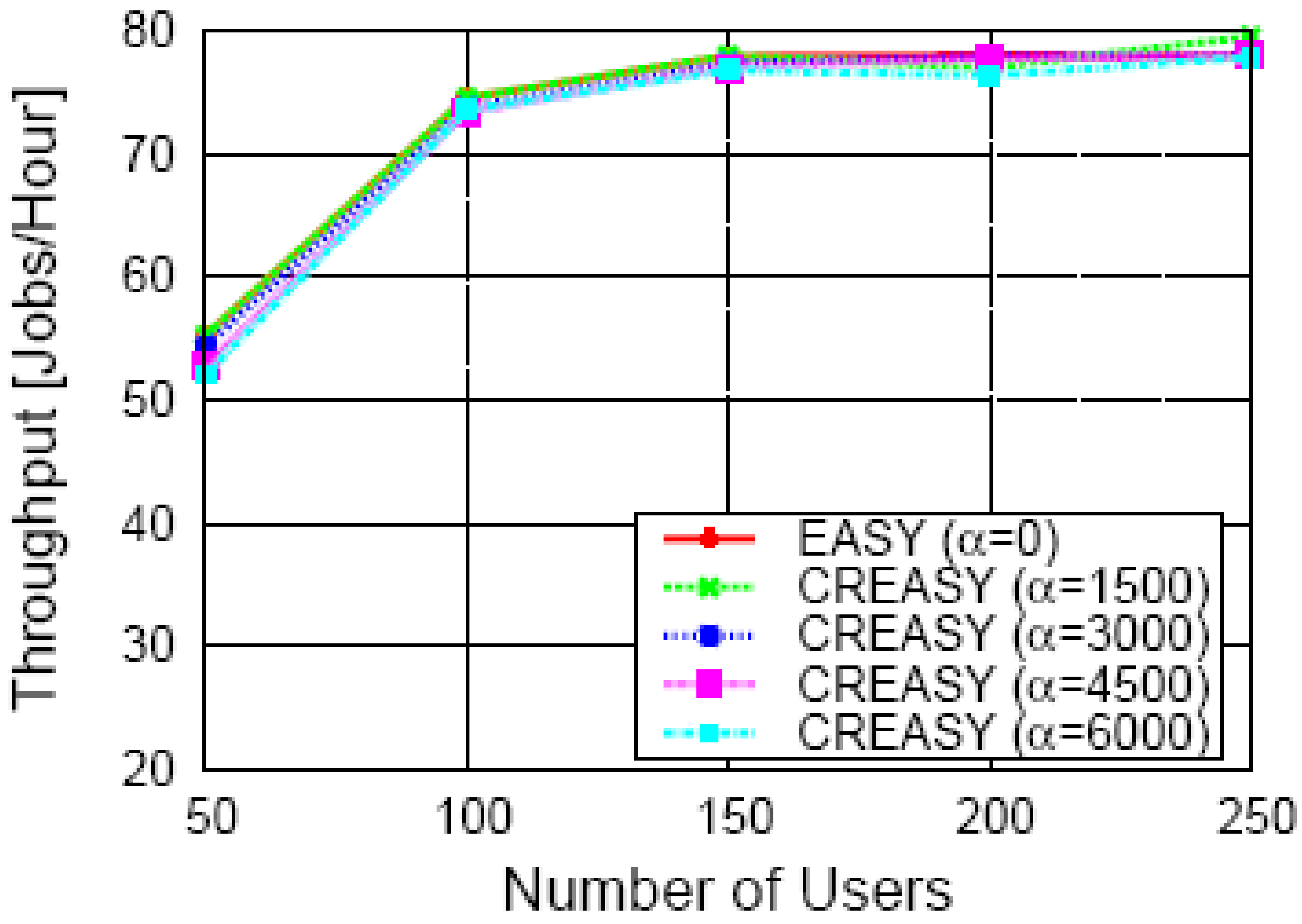
Understanding user behavior enables user cognizant system design

- Low response times lead to low think times
- Low think times mean longer sessions
- Longer sessions mean more jobs
- More jobs mean higher overall throughput

➡ Jobs should be prioritized by their expected response time

- The assumed user behavior
  - Probability to break session is proportional to response time
- The CREASY scheduler
  - Priority order for scheduling is proportional to expected response time = criticality

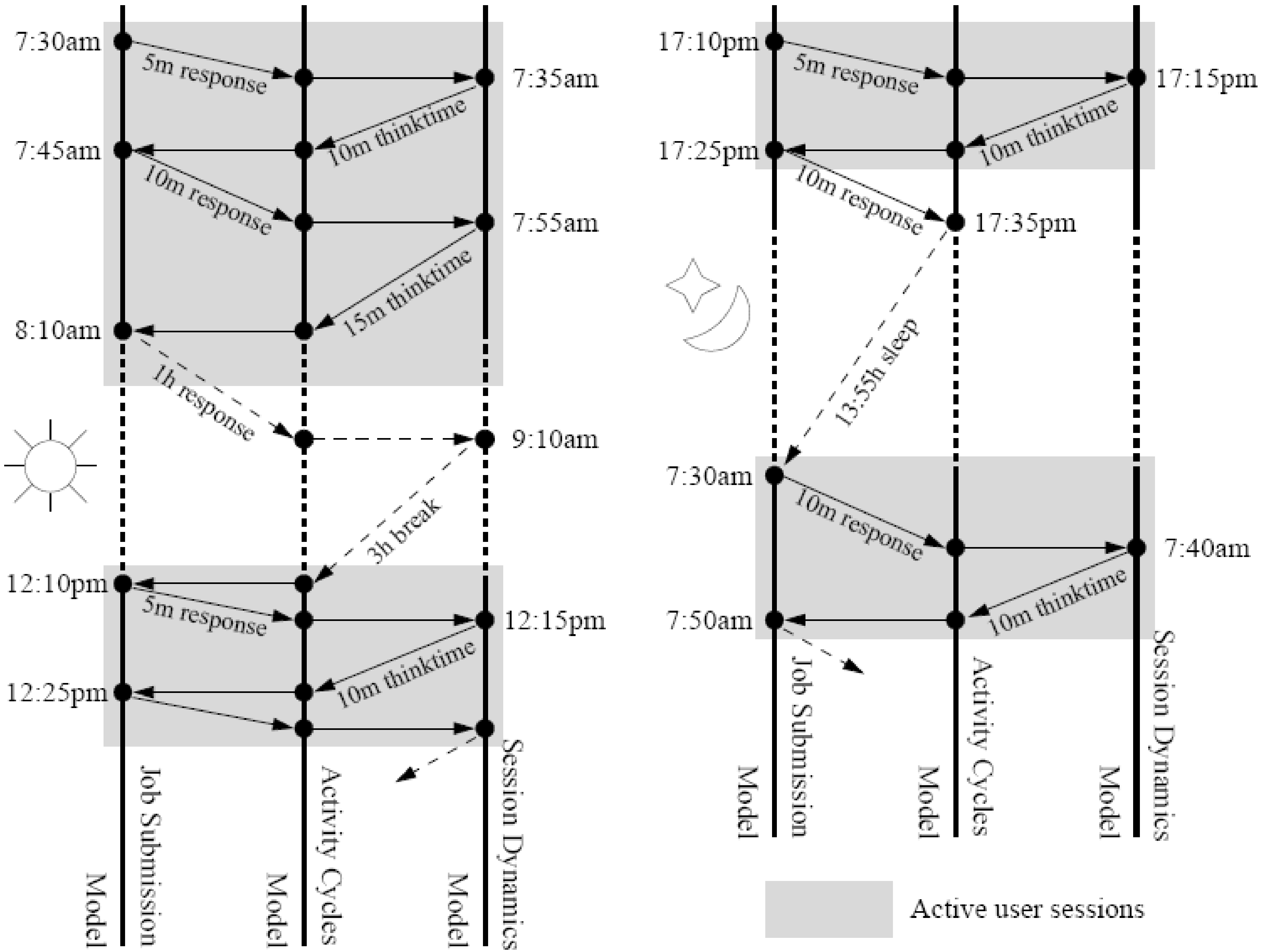
    (shorter response times get higher priority)
  - Similar to SJF, but short jobs that wait lose their advantage
  - This is combined with wait time to prevent starvation

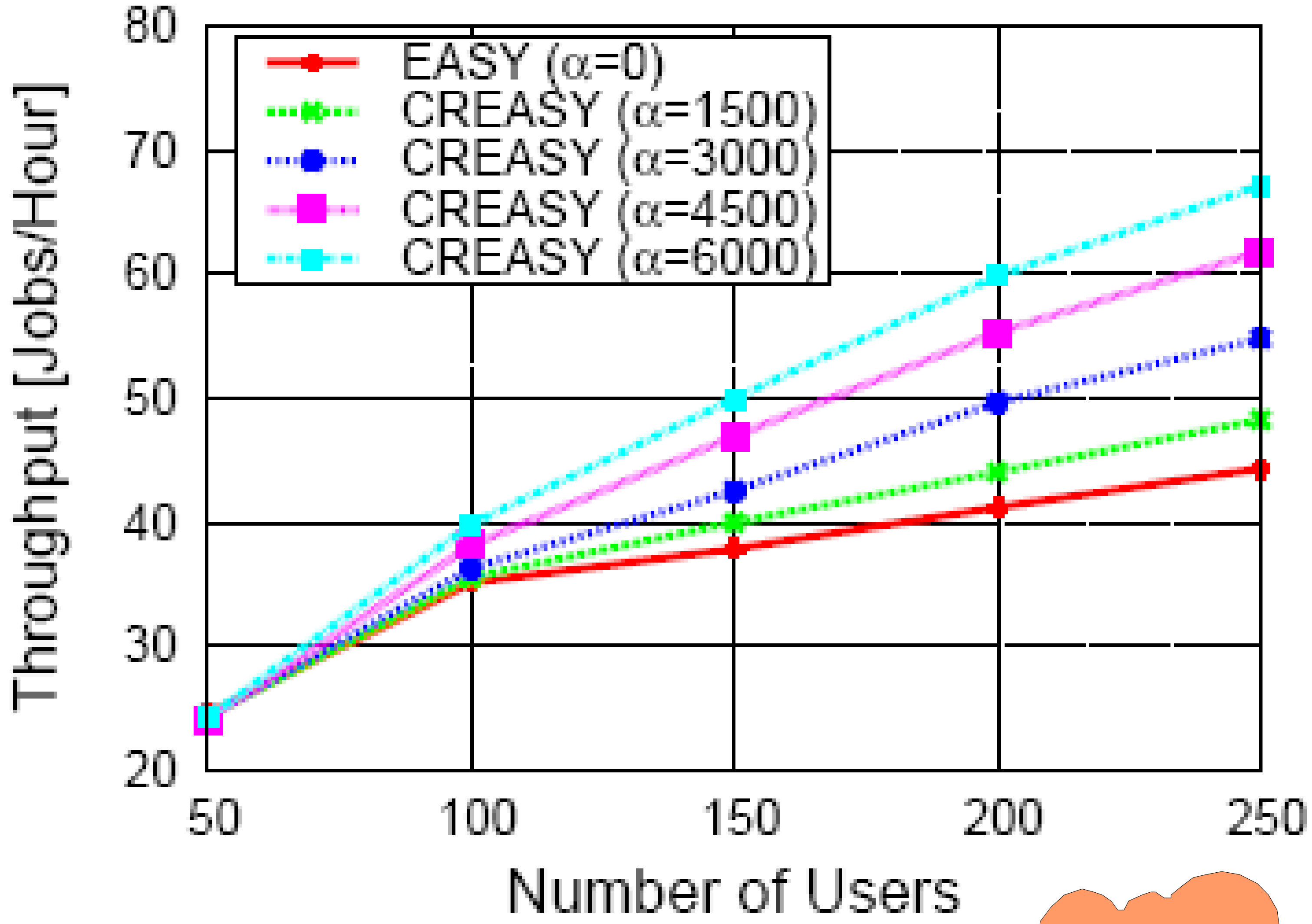


Simulation results: CREASY has no advantage over EASY

- The assumed user behavior
  - Probability to break session is proportional to response time
  - Users exhibit a daily cycle of activity and go to sleep at night
- The CREASY scheduler
  - Priority order for scheduling is proportional to expected response time = criticality

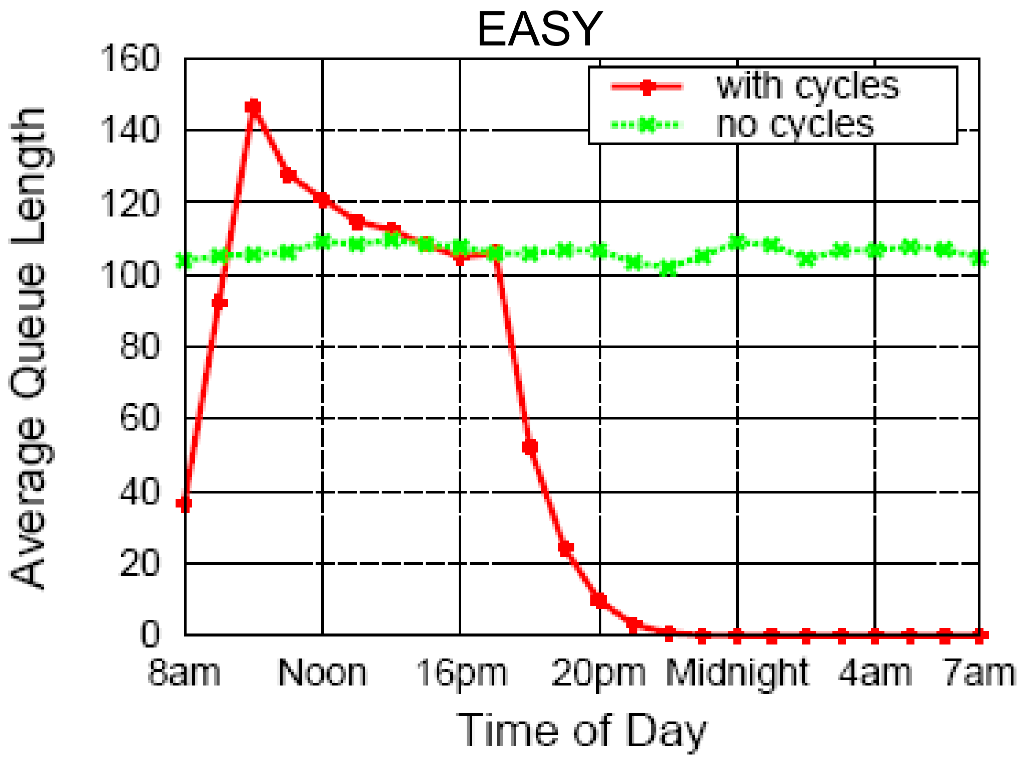
    (shorter response times get higher priority)
  - Similar to SJF, but short jobs that wait lose their advantage
  - This is combined with wait time to prevent starvation

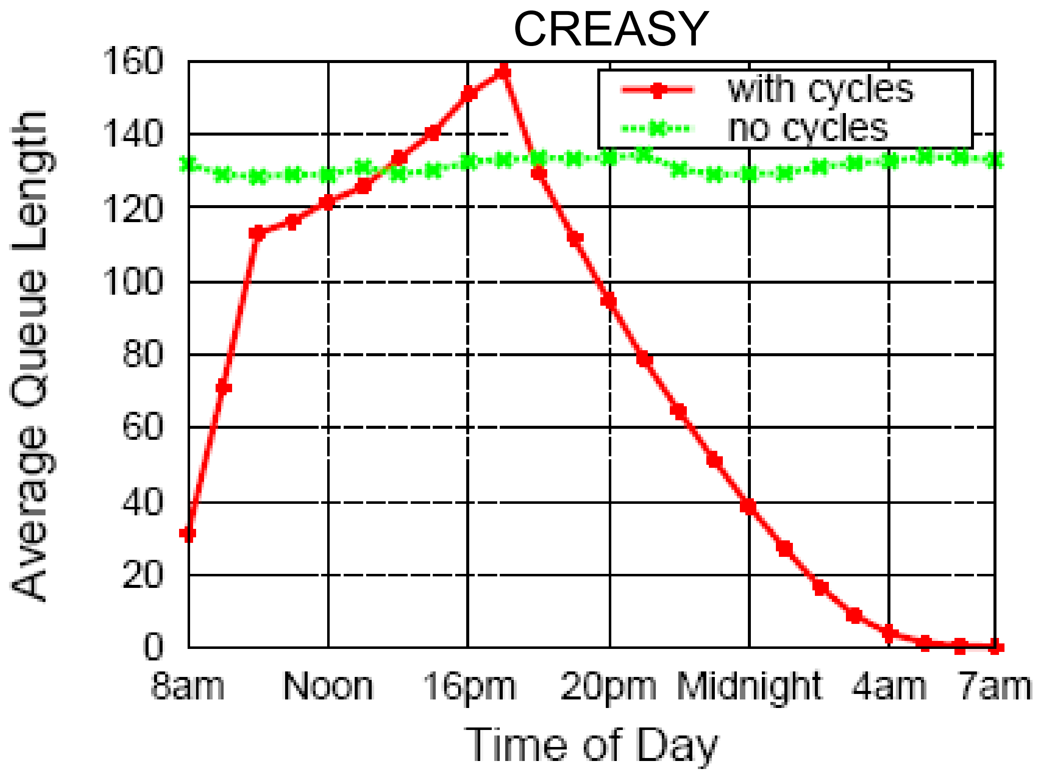

7:30am
5m response
7:35am
10m thinktime
7:45am
10m response
7:55am
15m thinktime
8:10am
1h response
9:10am
3h break
12:10pm
5m response
12:15pm
10m thinktime
12:25pm
Job Submission Model
Activity Cycles Model
Session Dynamics Model
17:10pm
5m response
17:15pm
10m thinktime
17:25pm
10m response
17:35pm
13:55h sleep
7:30am
10m response
7:40am
10m thinktime
7:50am
Job Submission Model
Activity Cycles Model
Session Dynamics Model
Active user sessions



?!?

Simulation results: with daily cycles C has an advantage that depends on the rela eight of the criticality consideration





- EASY is based on FCFS
- CREASY strongly prefers short response times
  - During the day, accumulates long jobs in queue
  - At night runs them using idle resources, thus increasing throughput and utilization
  - Can't do this if there is no idle time at night!

Summary:

Feedback is an important aspect of computer workloads

It can have profound effects on system behavior and should not be ignored in evaluations

Evaluations based on a generative workload model are more reliable than evaluations based on replaying a job trace

Model should be conservative and include all workload features

Mini-project:

- Define criteria for extracting feedback from logs
  - Each job may depend on a previous job by the same user
  - But which one?
  - And when should it be considered to be independent?
- Write program to add dependency information to job logs
- Evaluate by comparing distributions of session length etc.