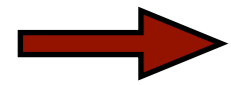


Cluster scheduling for explicitly-speculative tasks

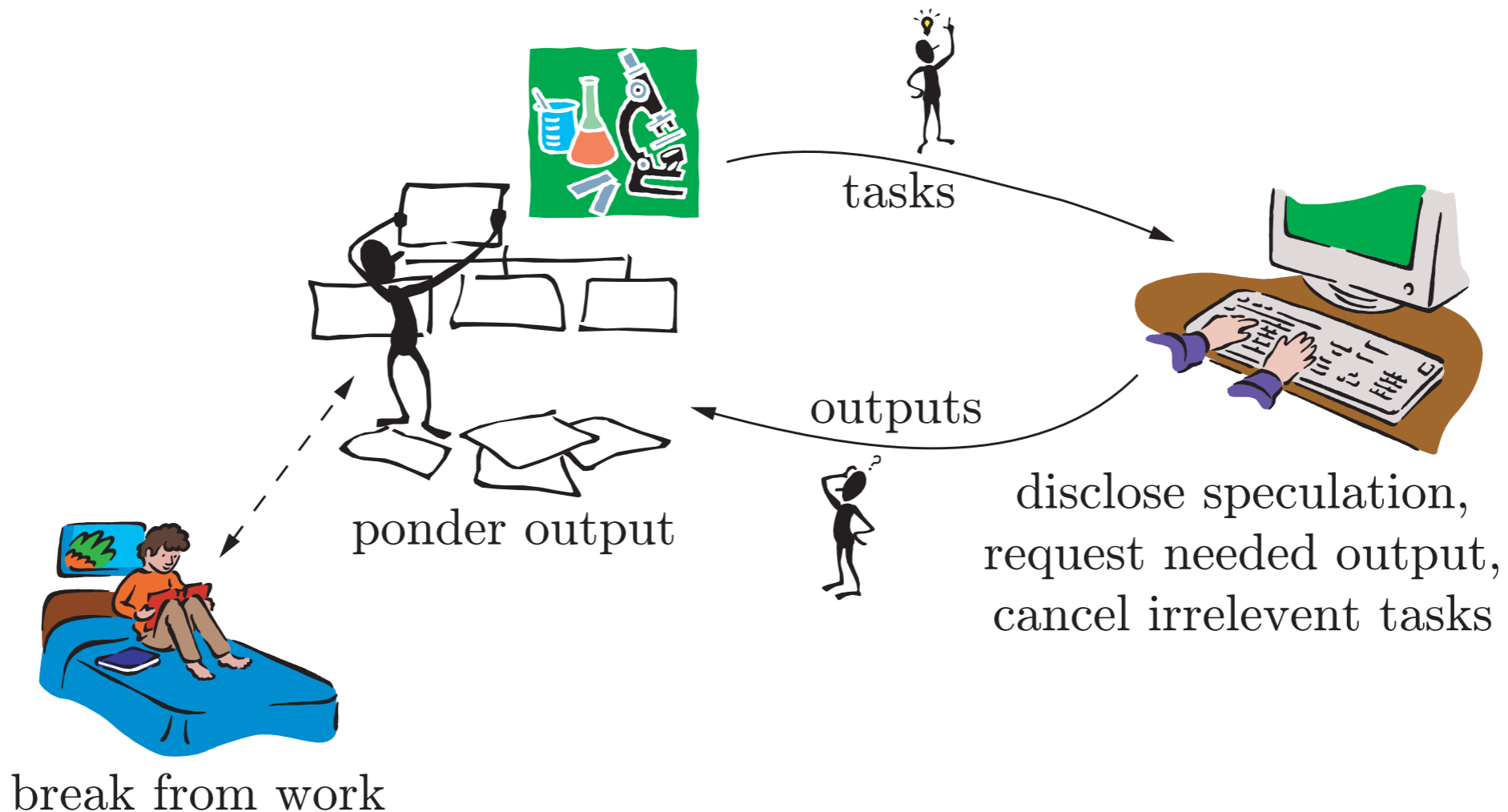
David Petrou

Agenda



- Motivation & thesis in a nutshell
- Target applications
- Related work
- **Batchactive** scheduling
- Experimental design
- Think time
- Speculation
- Incentive pricing
- Proposed deployment
- Contributions & conclusions

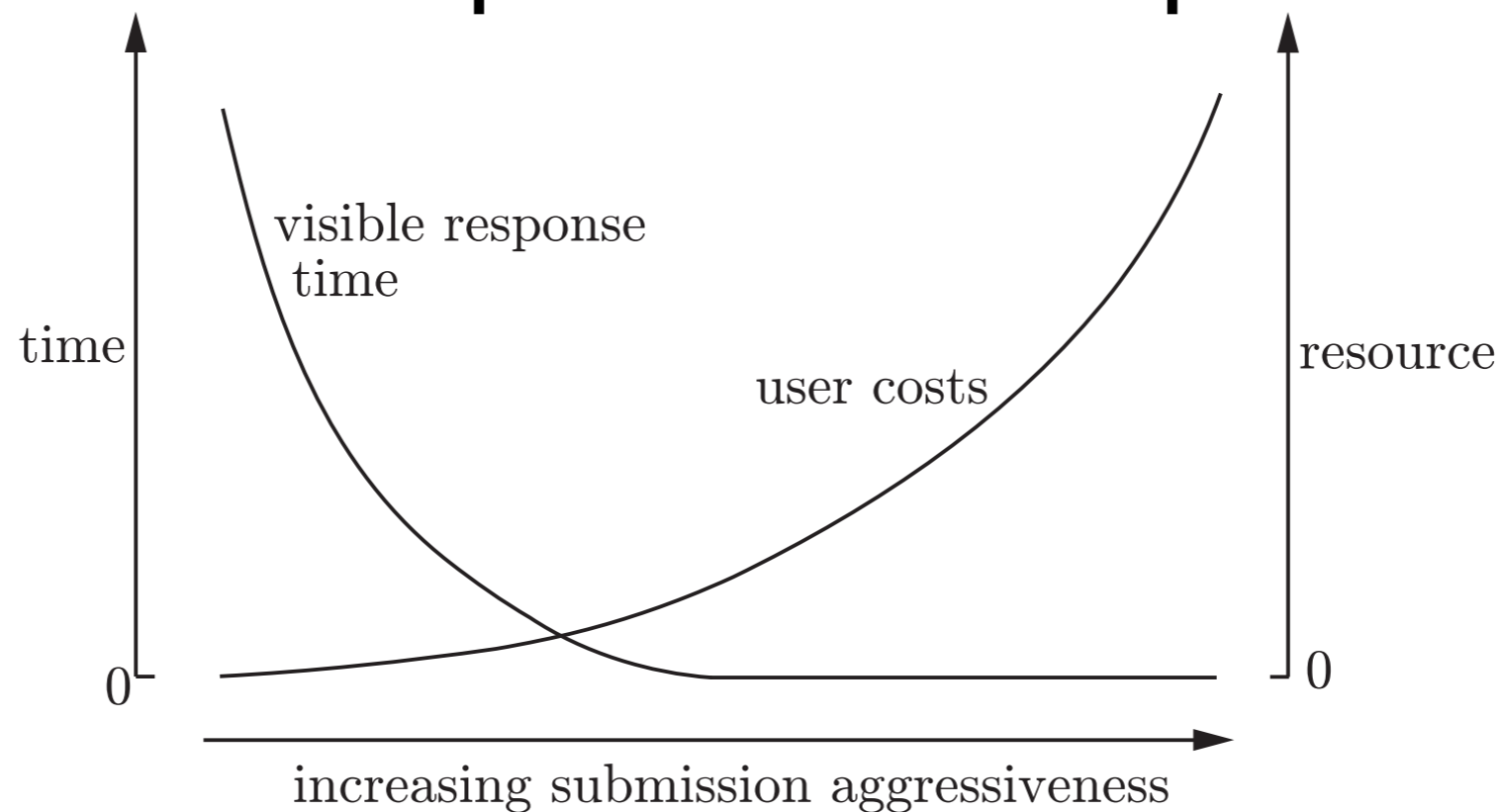
Speculative user behavior



- Exploratory searches, parameter studies
- Users desire fast response
- Resource provider desires high utilization

Problem: schedulers do not exploit chains of potential work

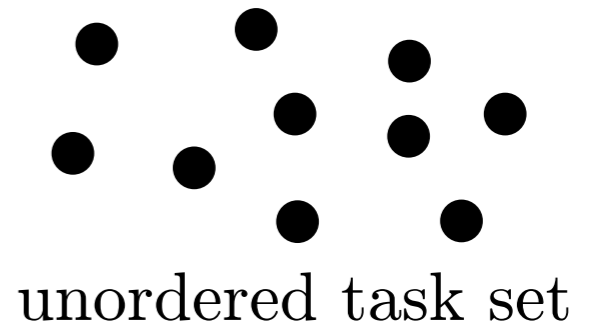
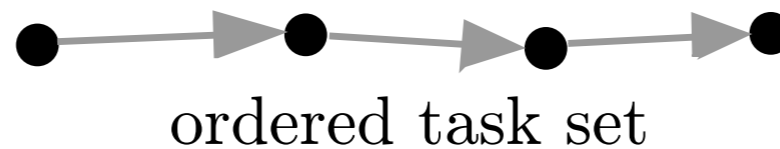
- Manual speculation wastes resources & leads to poor visible response time



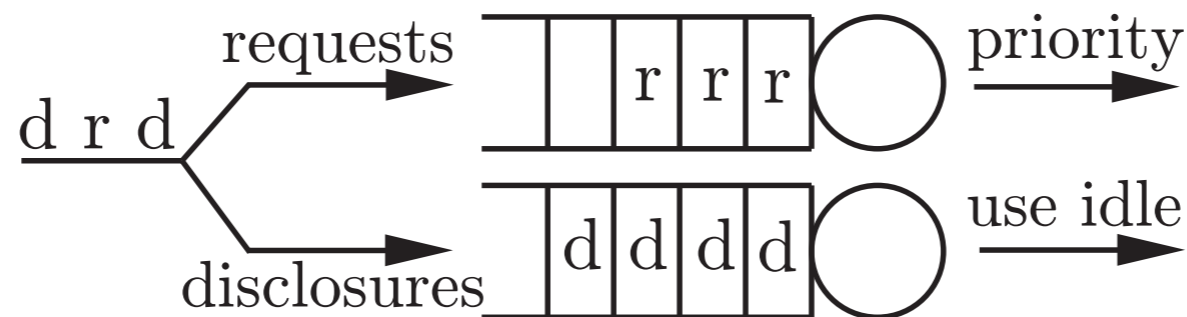
- User cannot decide what speculation to issue; paying more, waiting longer

Solution: batchactive scheduling

- Users disclose speculative tasks



- Speculation-aware scheduling
 - direct support for exploratory searches
 - exploits knowledge of what is not yet needed
- Examine perf. using simplest spec.-aware sched.:



- At most **1/2x** mean visible response time over **20%** of simulated scenarios

Thesis

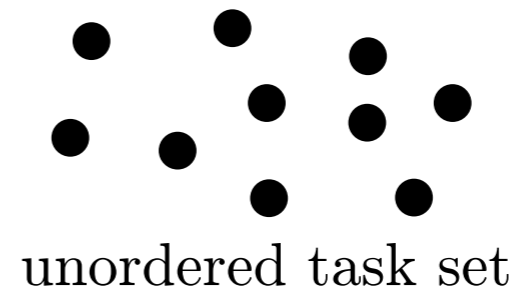
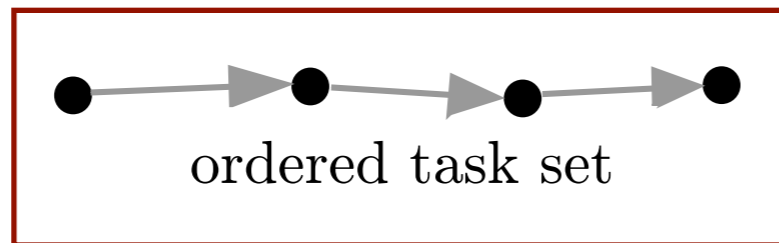
- A multiuser process scheduler informed of which tasks are speculative can provide better time- and cost-based metrics for users and resource providers:
 - speculative tasks are poorly exploited by existing schedulers – no think time
 - batchactive scheduling significantly reduces visible response time for exploratory searches
 - an incentive pricing mechanism encourages use, improves user costs, & sometimes improves server revenue

Agenda

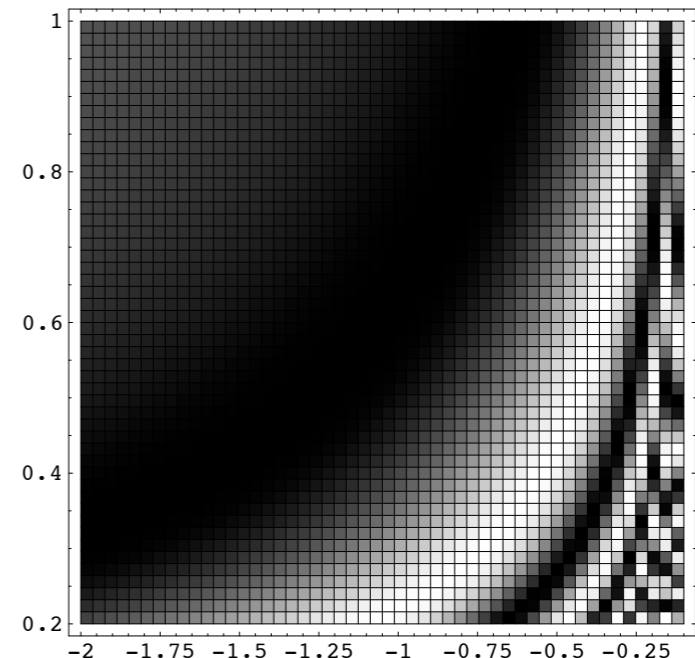
- Motivation & thesis in a nutshell
- ➔ ● Target applications
- Related work
- Batchactive scheduling
- Experimental design
- Think time
- Speculation
- Incentive pricing
- Proposed deployment
- Contributions & conclusions

Speculative search

- Value of user time increasing & cost of processing decreasing → people willing to put more effort into using resources speculatively
- Manual or automatic **task set** specification




- explore high dimensional spaces, improve resolution



Exploratory searches, sequential tasks, parameter studies

- **Bioinformatics:** quickly identify non-matches in DNA with BLAST, increase accuracy for promising paths; protein folding
- **Computer animation:** speculatively render scenes, discard and retry if can be improved
- **Design by computer:** simulations search spaces of network behavior, cache arch., ...
- Target architecture: **clusters**
- Scope: **non-parallel tasks**

Agenda

- Motivation & thesis in a nutshell
- Target applications
-  ● Related work
- Batchactive scheduling
- Experimental design
- Think time
- Speculation
- Incentive pricing
- Proposed deployment
- Contributions & conclusions

Related work

- **Tasks:** optim. make [Bubenik89], parallel worlds [Sun99]
- **Proc:** spec. instructions until branch resolved [Smith88]
- **I/O:** disclosing reads [Patterson95], automatic [Chang99]
- **Network:** web prefetching [Padmanabhan96,Steere97], parallel apps on slower grid networks [Chrisochoides03]
- **Languages:** spec. Multilisp [Osborne90], BaLinda [Yee93]
- **DBs:** query during think time [Polyzotis03]
- **Think time exposed for pervasive speculative techniques**
- **Exploiting idle time:** TCP-Nice [Venkataramani02], Ms. Manners [Douceur99], preempt overhead [Eggert04]

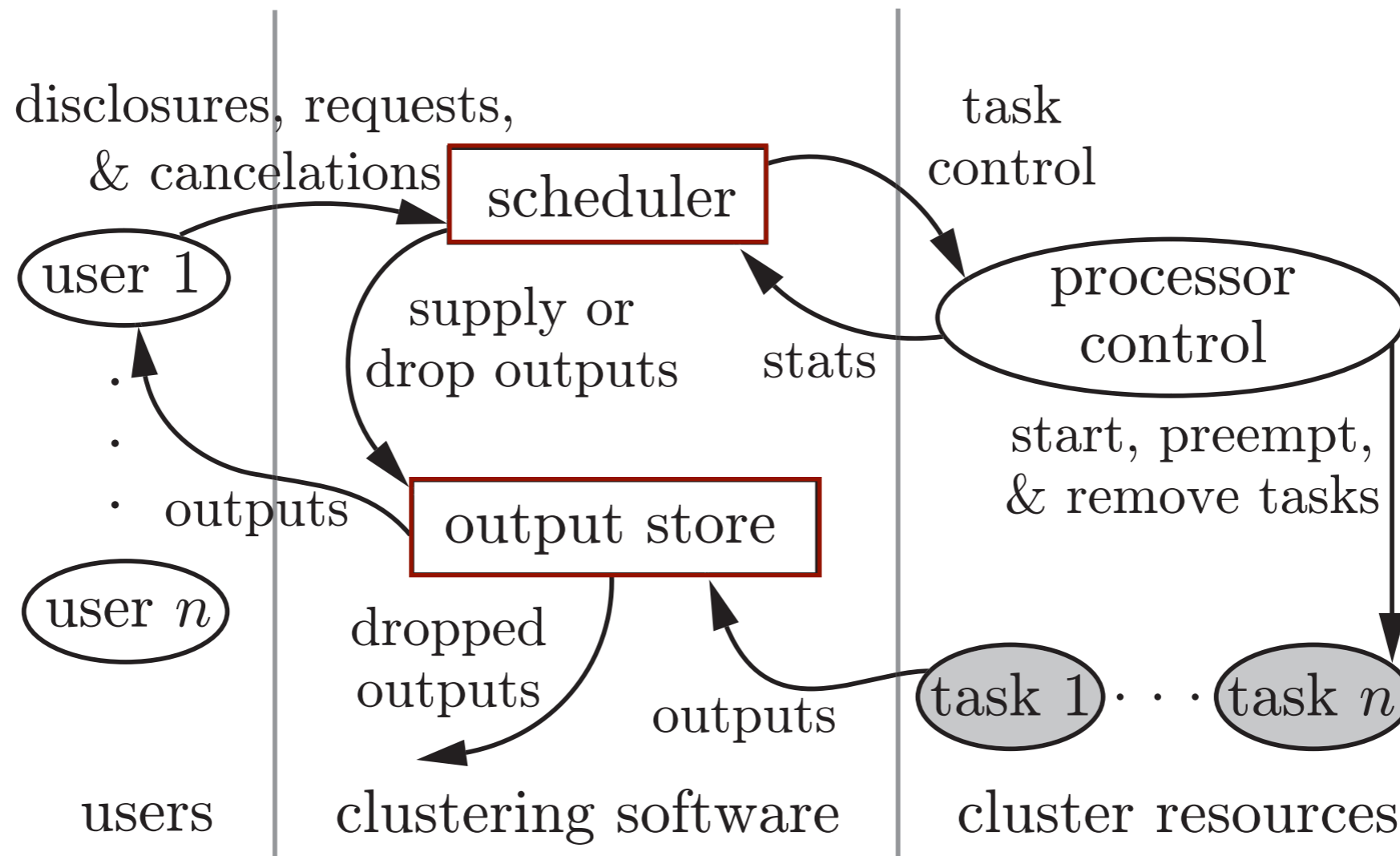
- Batchactive scheduling operates at the **task granularity** for **shared clusters**
 - **Manually-disclosed chains of future work**

Agenda

- Motivation & thesis in a nutshell
- Target applications
- Related work
- ● Batchactive scheduling
- Experimental design
- Think time
- Speculation
- Incentive pricing
- Proposed deployment
- Contributions & conclusions

Batchactive cluster

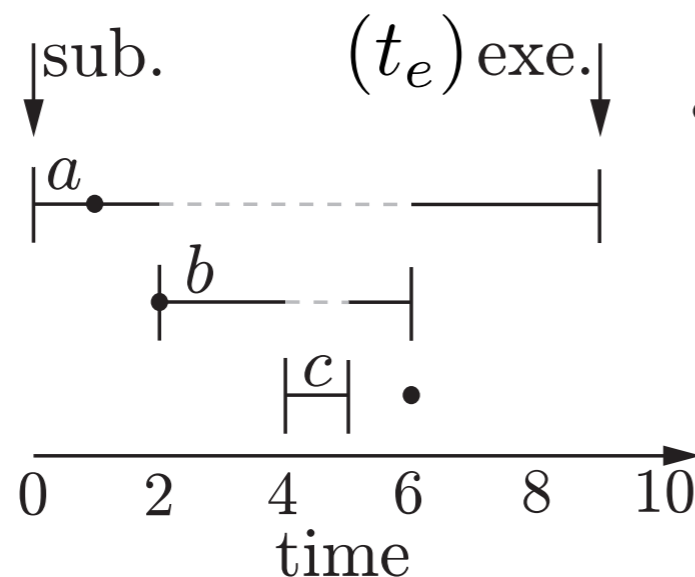
- Users distinguish between tasks needed now (**requests**) and potentially future needed tasks (**disclosures**)



Time accounting for chains

- Users desire the time between needing and receiving task output to be minimized
- Traditional response time confuses time of submission and time of need
- New metric: **visible response time**

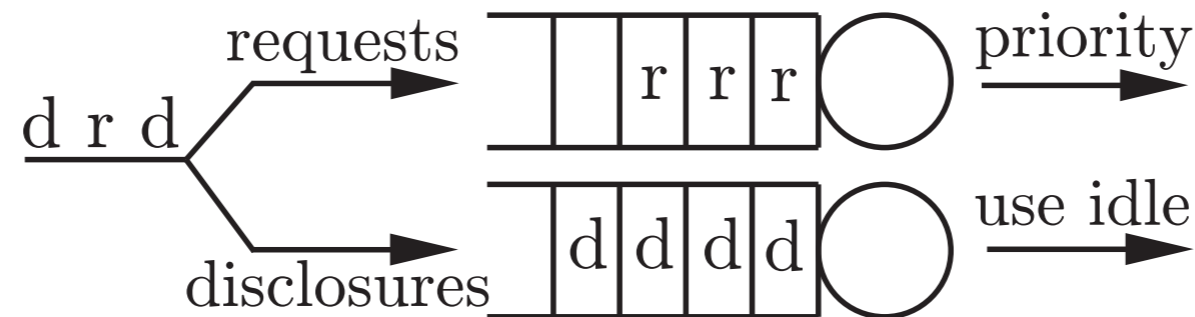
$$V_a^{\text{resp}} \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } t_n > t_e, \\ t_e - t_n & \text{if } t_n \leq t_e. \end{cases}$$



| | <i>a</i> | <i>b</i> | <i>c</i> |
|-----------------|----------|----------|----------|
| service time | 5 | 3 | 1 |
| vis. resp. time | 8 | 4 | 0 |

Batchactive scheduling

- Uses knowledge of what is speculative
- Start with a simple **two-tiered** scheduler to demonstrate overall approach



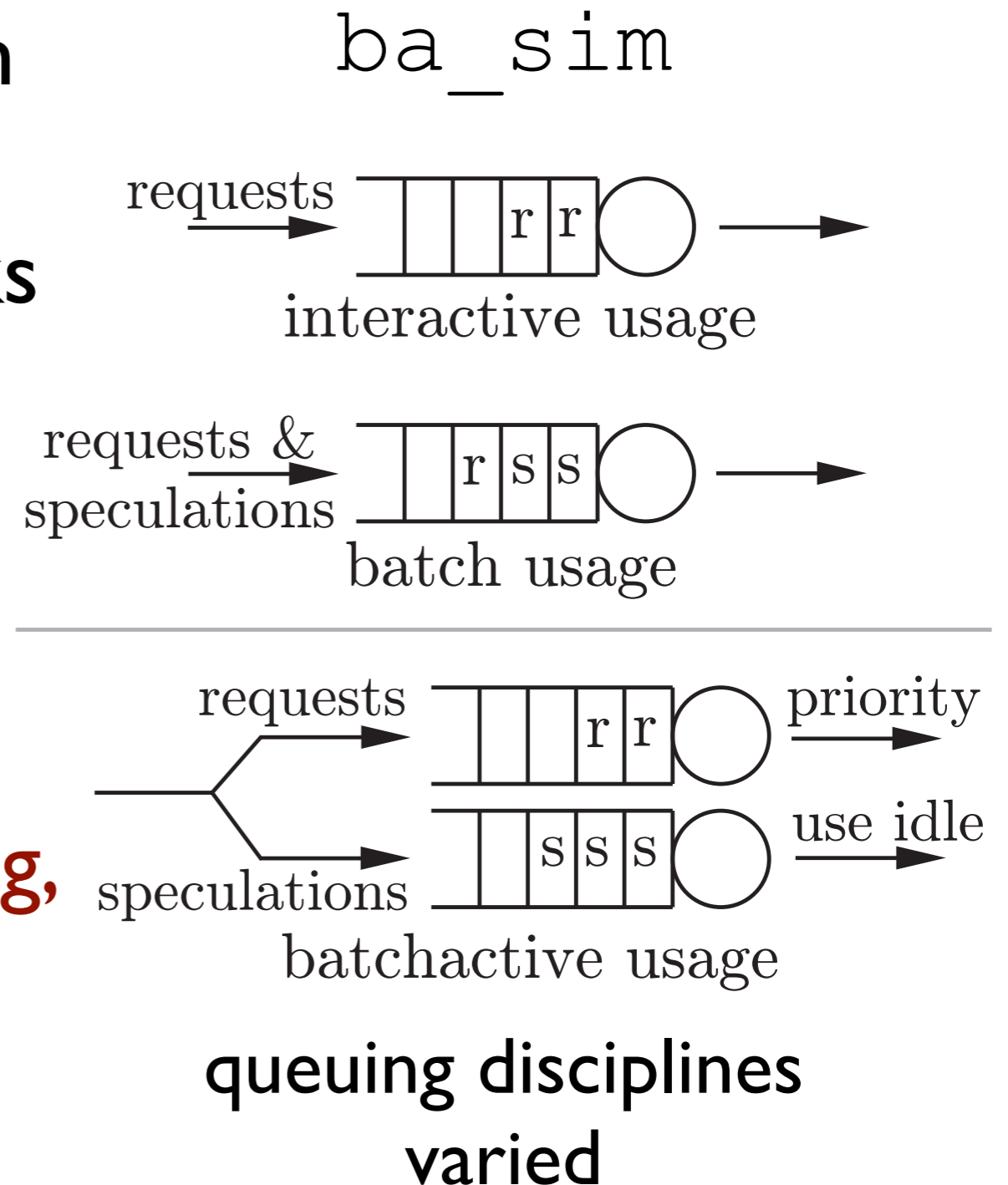
- Req. queue policies: FCFS, SRPT, usage-based
- New policy for disclosed tasks
 - **highest request probability (HRP)**
- Information / performance tradeoff

Agenda


- Motivation & thesis in a nutshell
- Target applications
- Related work
- Batchactive scheduling
- ➔ ● Experimental design
- Think time
- Speculation
- Incentive pricing
- Proposed deployment
- Contributions & conclusions

Experimental design

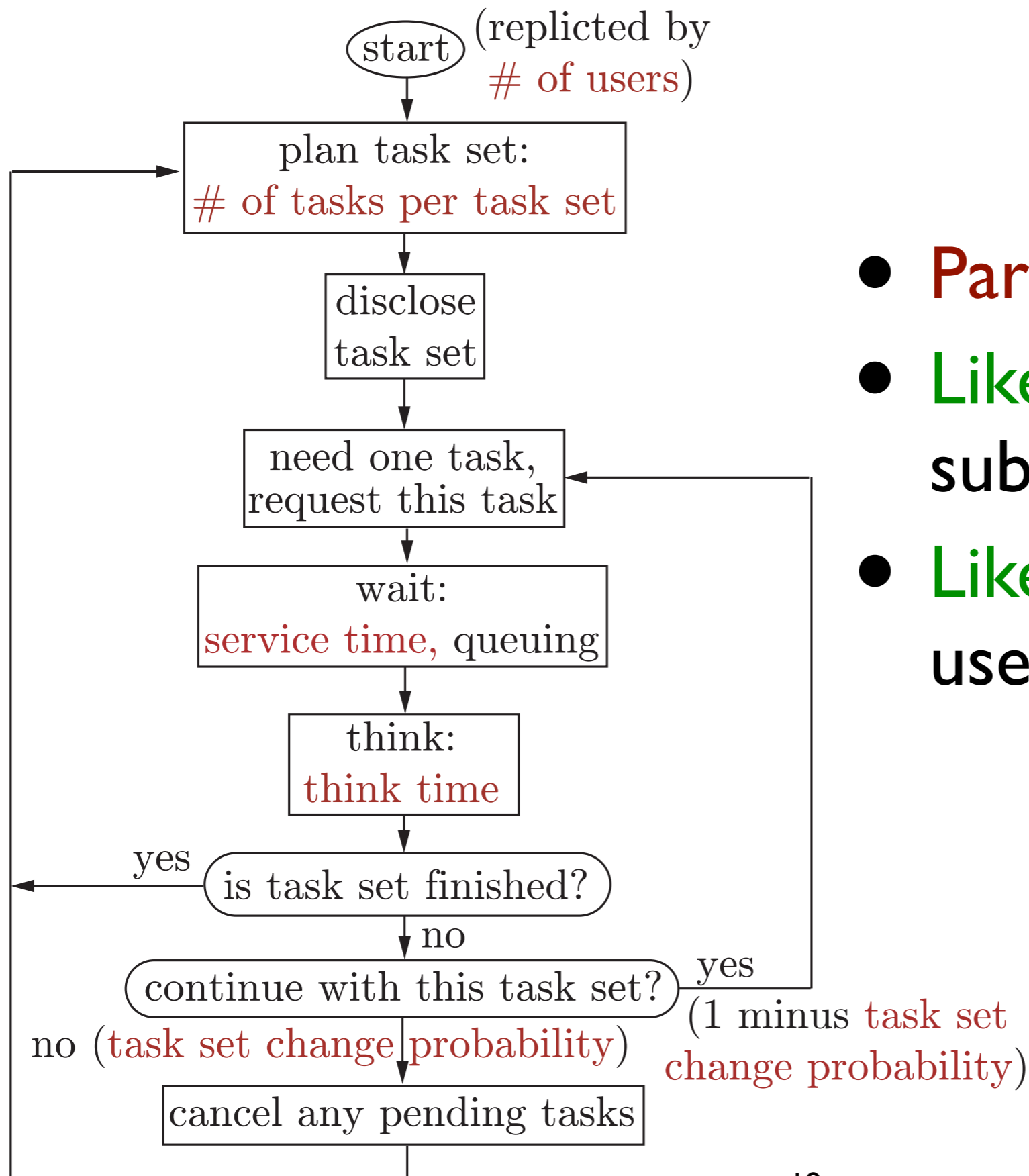
- Compare metrics between traditional and batchactive schedulers over two-weeks
- Users with a traditional scheduler **request** tasks in **batches** or **interactively** (one at a time)
- Batch: **performance-seeking**, interactive: **risk-averse**
- Single-server model



Simulation parameters

- Explore large space of user & task behavior
- Five simple but powerful knobs
 - service time (task size)
 - think time (when the next task is needed)
 - number of tasks per task set  per-user degree of speculation
 - prob. of canceling & issuing a new task set
 - number of users (concurrency)
- Batchactive scheduling gives **much better performance** for some parts of this space and is **never appreciably worse**

Batchactive user model



- Parameters in red
- Like batch: chains are submitted at once
- Like interactive: tasks users need are exposed

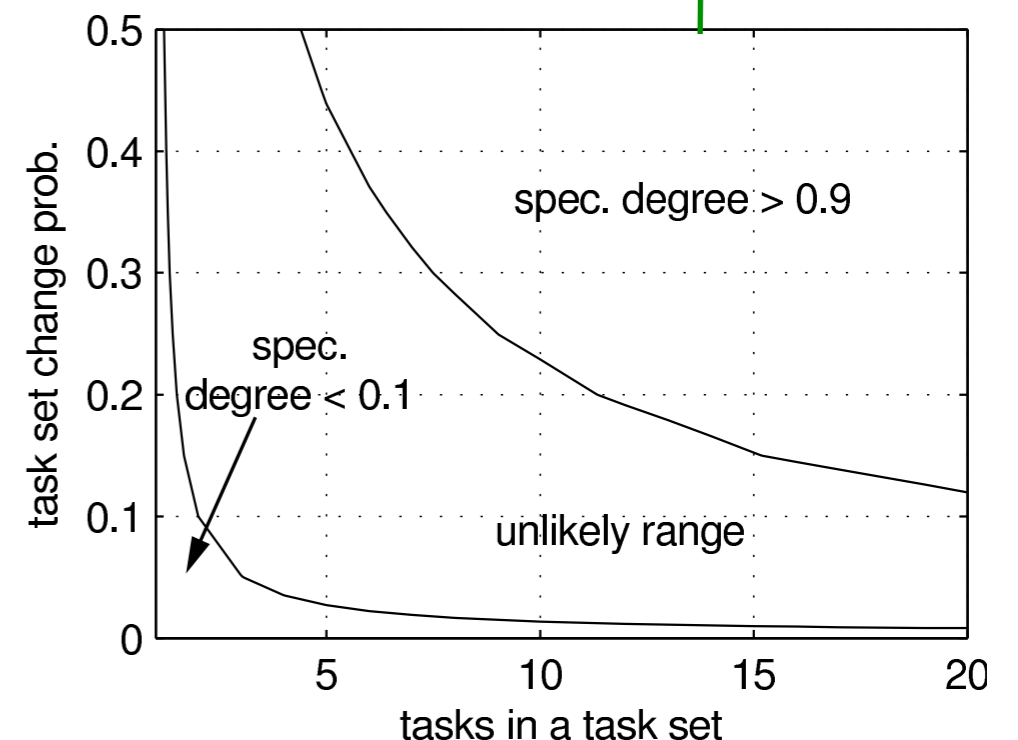
Workload space characterization

- Actual user behavior unknown
- **Goal:** characterize behavior over wide range, even no speculation & low think time

5,400 scenarios

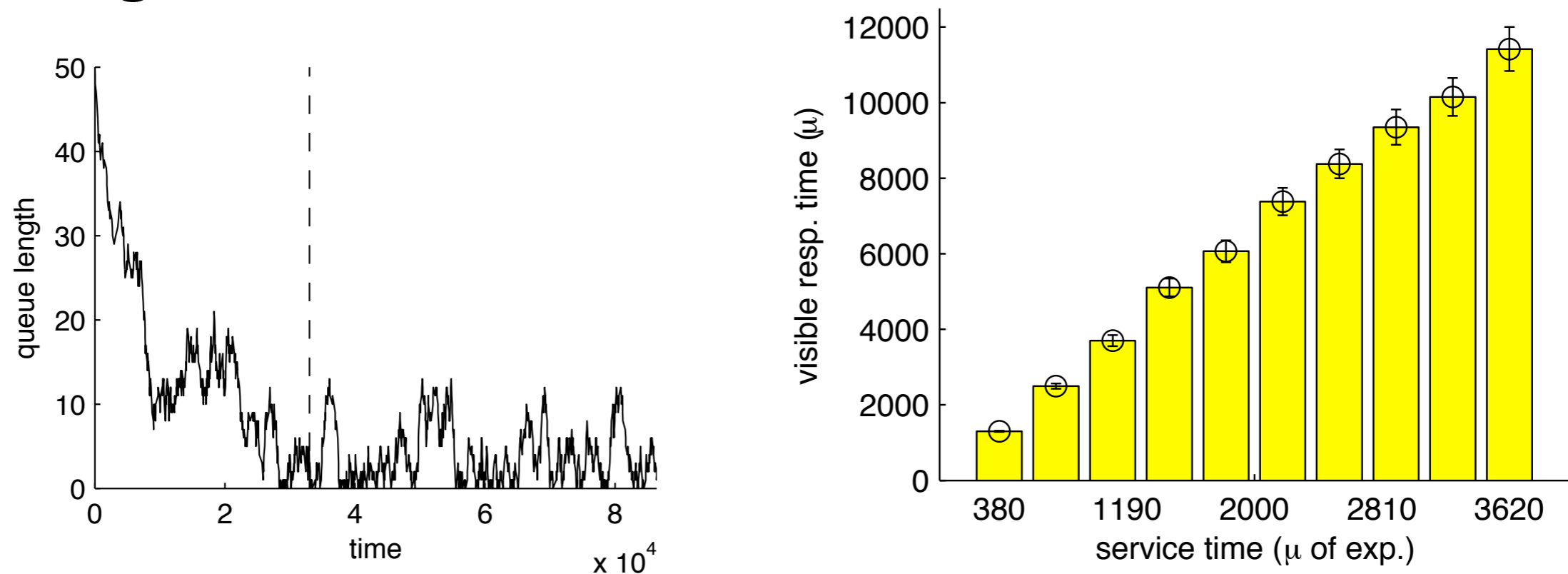
| parameter | range |
|-------------------------|--------------------------|
| service time | 20 s–1 hr (exponential) |
| think time | 20 s–5 hr (exponential) |
| # of tasks per task set | 1 to 1–21 (uniform) |
| task set change prob. | 0.0 to 0.0–0.4 (uniform) |
| # of users | 1–16 |

- Includes BLAST, rendering
- Fast & slow contemplation
- Useful numbers of tasks
- Low to high cancelation
- A range of concurrency



Sim. details & model verification

- Omitted warmup period and statistical significance of results

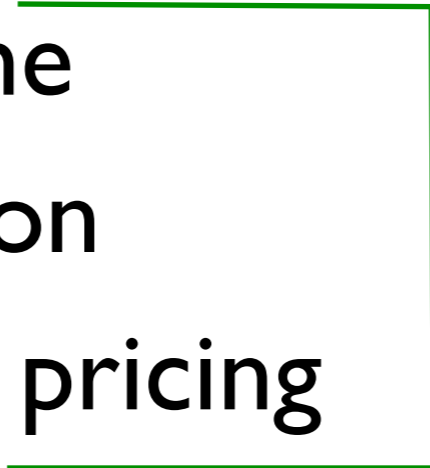
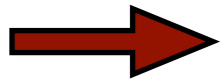


- Degenerate tests, software engineering, hand-checked small runs
- Non-speculative verif. using operational laws

| | theoretical | simulation |
|-----------------|-------------|------------|
| mean resp. time | 1026.710 | 1012.618 |
| load | 0.918618 | 0.917004 |

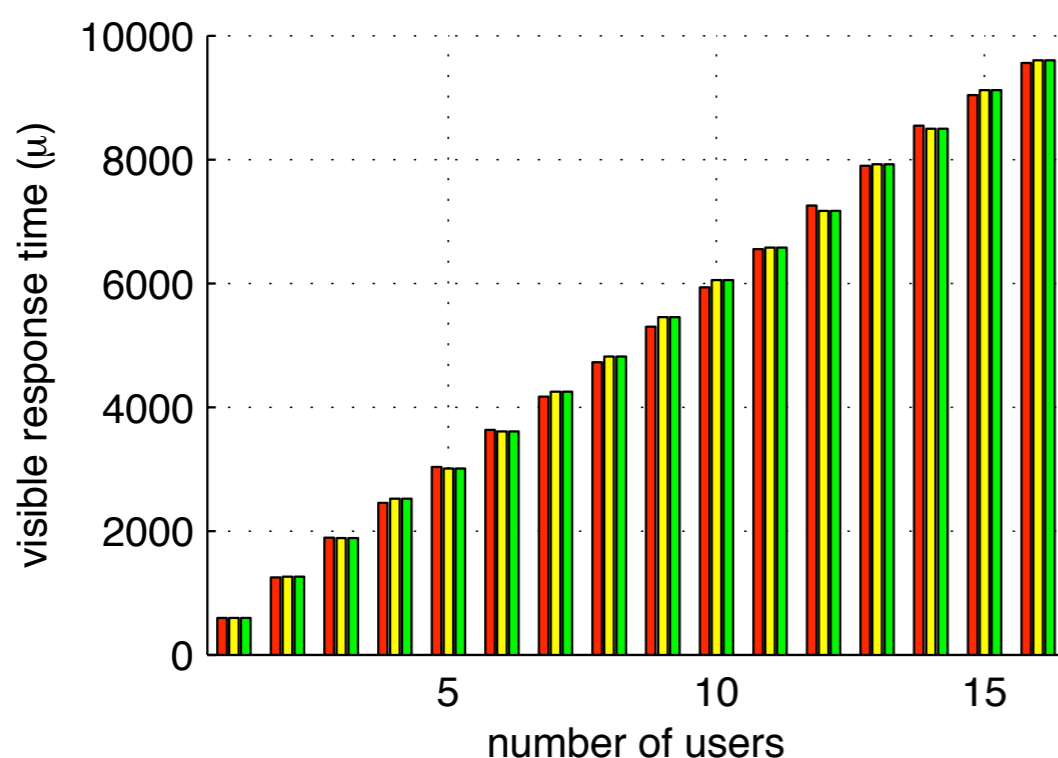
Agenda

- Motivation & thesis in a nutshell
- Target applications
- Related work
- Batchactive scheduling
- Experimental design
- Think time
- Speculation
- Incentive pricing
- Proposed deployment
- Contributions & conclusions



The need for think time

- Other speculative systems know think time: TIP
- Might not be a user who thinks
- Delay before user is ready for next output should be exposed to task scheds [Feitelson97]
- Degenerate case: equal perf. without think time

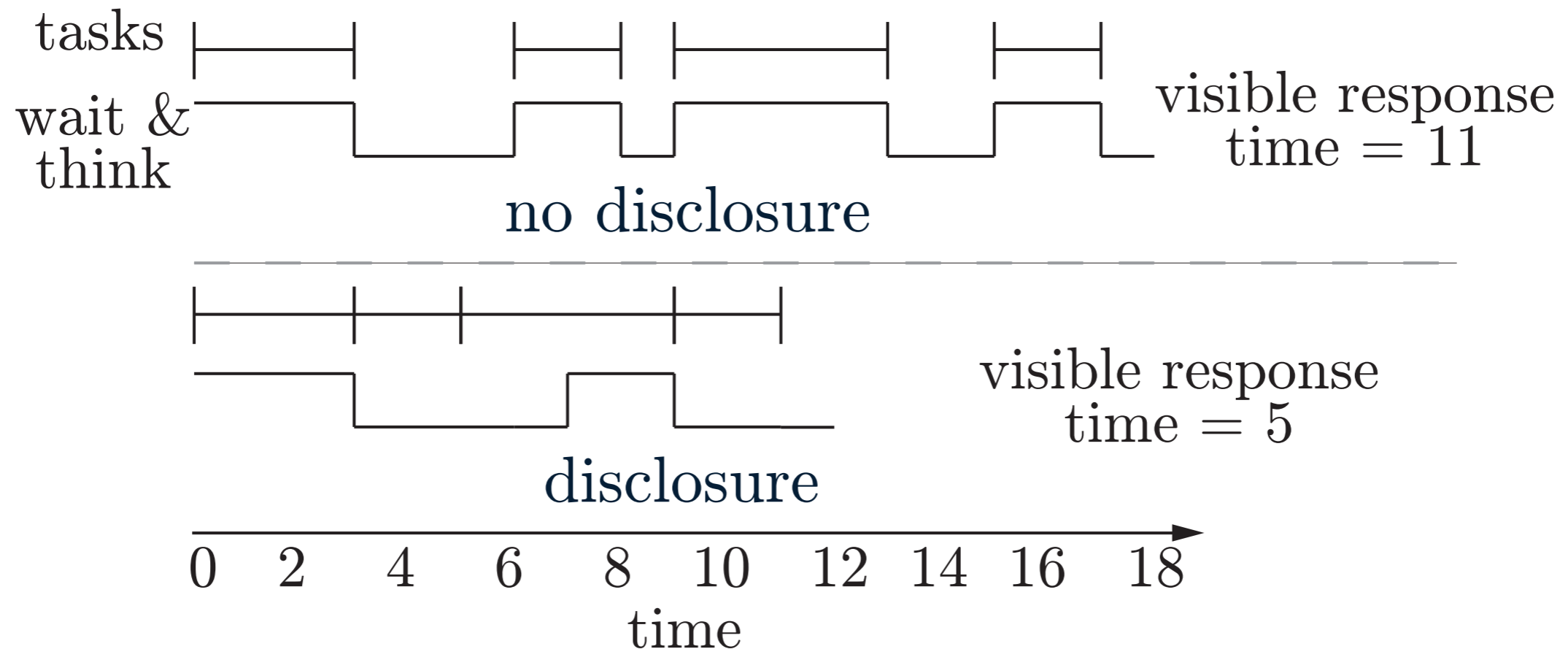


■ FCFS, batch
■ FCFS, inter.
■ FCFS x FCFS

requested x disclosed queue polices ←

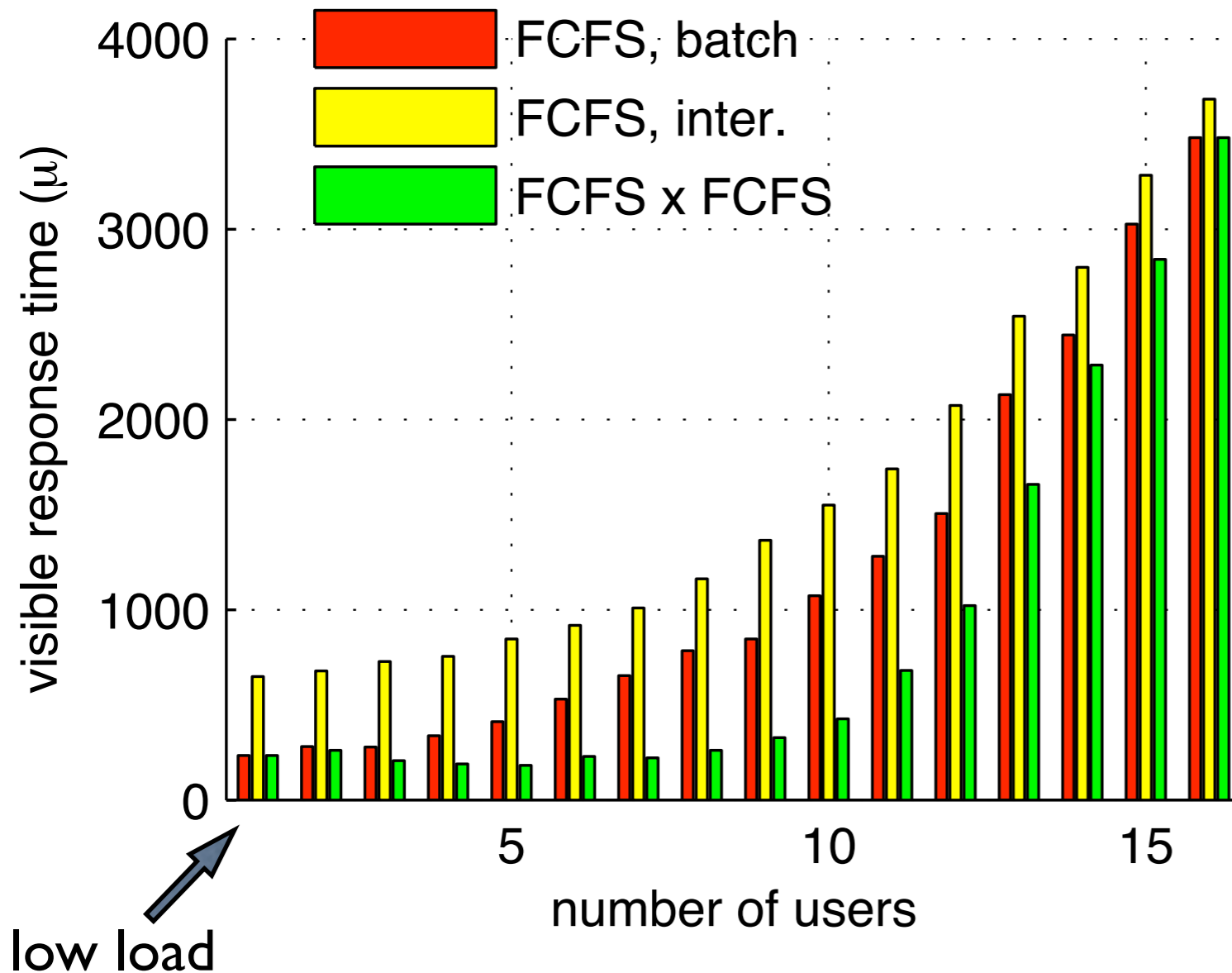
| parameter | setting |
|---------------------|-------------------|
| t.s. change prob. | 0.0 to 0.2 (uni.) |
| # of tasks per t.s. | 1 to 15 (uni.) |
| service time (s) | 600 (exp.) |
| think time (s) | 0.0 to 0.0 (uni.) |

Effect of disclosure and think time



- Disclosure enables the pipelining of task execution and user think time

The benefit of disclosing tasks even without speculation



| parameter | setting |
|---------------------|-------------------|
| t.s. change prob. | 0.0 to 0.0 (uni.) |
| # of tasks per t.s. | 1 to 15 (uni.) |
| service time (s) | 600 (exp.) |
| think time (s) | 6,000 (exp.) |

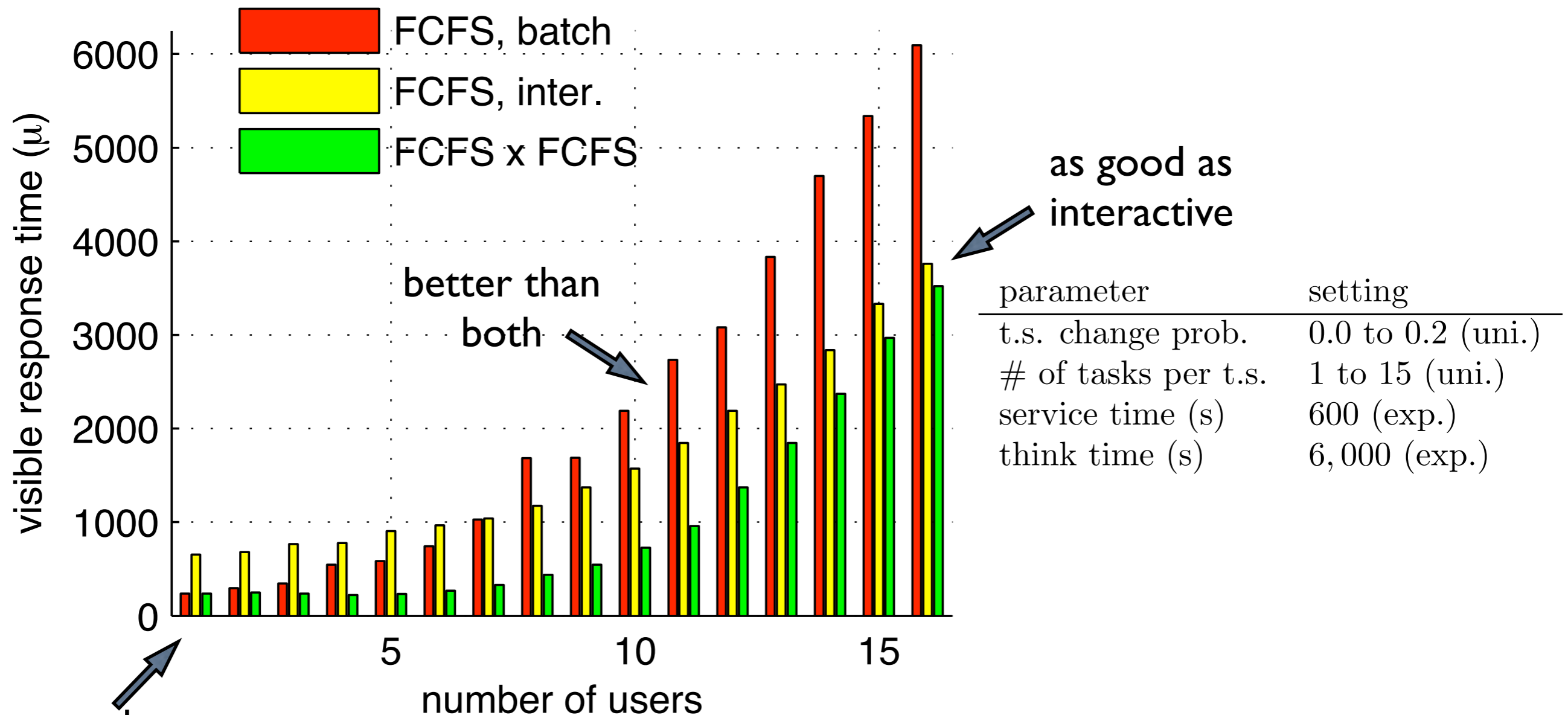
$$V_a^{\text{resp}} \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } t_n > t_e, \\ t_e - t_n & \text{if } t_n \leq t_e. \end{cases}$$

- Across a mid-range of users, batchactive wins

Agenda

- Motivation & thesis in a nutshell
- Target applications
- Related work
- Batchactive scheduling
- Experimental design
- Think time
- ➔ ● Speculation
- Incentive pricing
- Proposed deployment
- Contributions & conclusions

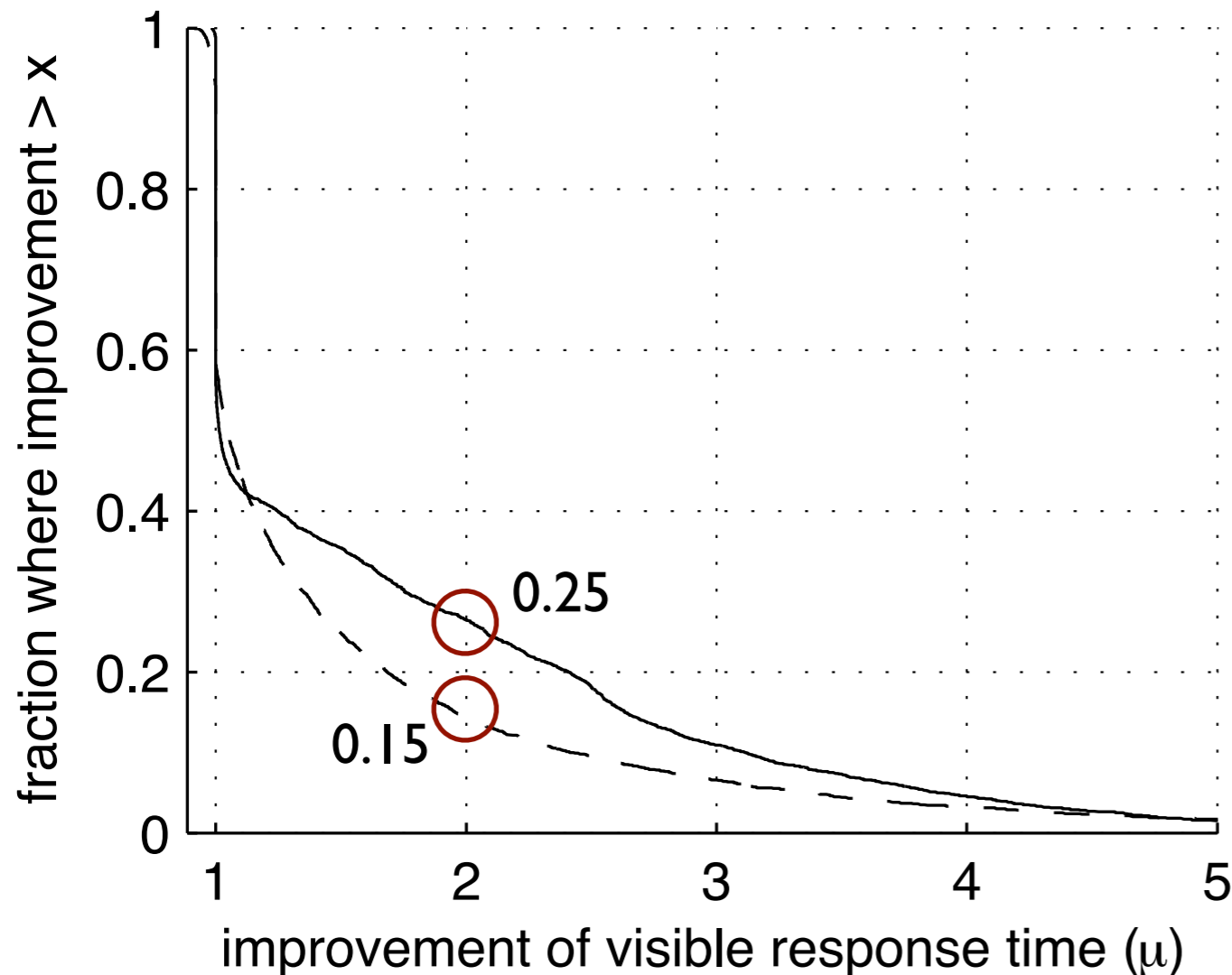
Generalized best case for speculation



as good
as batch

- Better improvement over FCFS, batch
- Batchactive adapts across load
- Visible throughput simultaneously improves

FCFS x FCFS improvements for mean visible response time



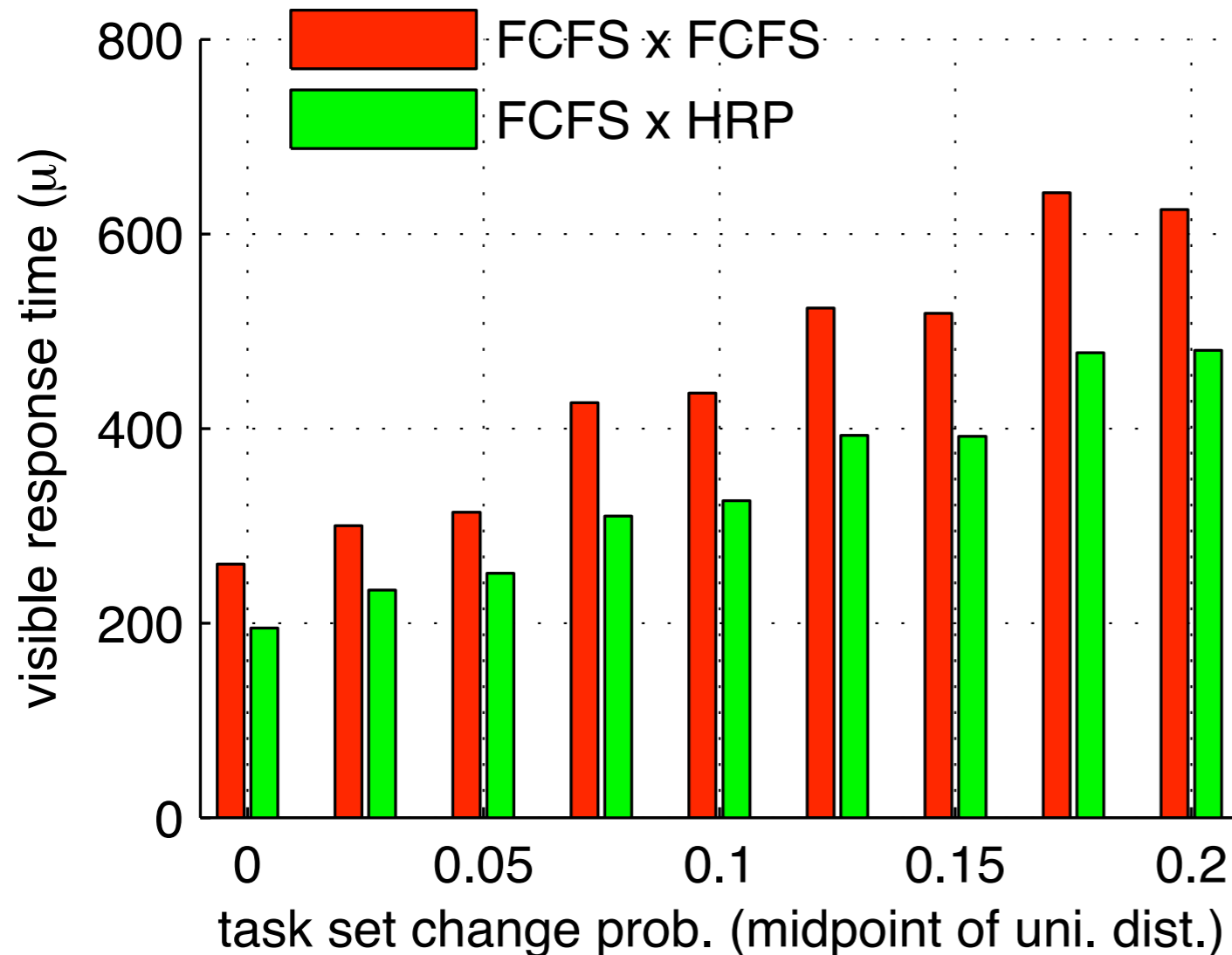
— — — over FCFS, batch
 ————— over FCFS, inter.

| param | range |
|---------------------|-----------------------|
| # of users | 1 to 16 |
| t.s. change prob. | 0.0 to 0.0–0.4 (uni.) |
| # of tasks per t.s. | 1 to 1–21 (uni.) |
| service time | 20 to 3,620 (exp.) |
| think time | 20 to 18,020 (exp.) |

$$\text{improvement} = \frac{\text{(old \#)}}{\text{(new \#)}}$$

- Batchactive mean visible response time is at most $1/2x$ for 15% and 25% of the 5,400 cases against batch and interactive, respect.

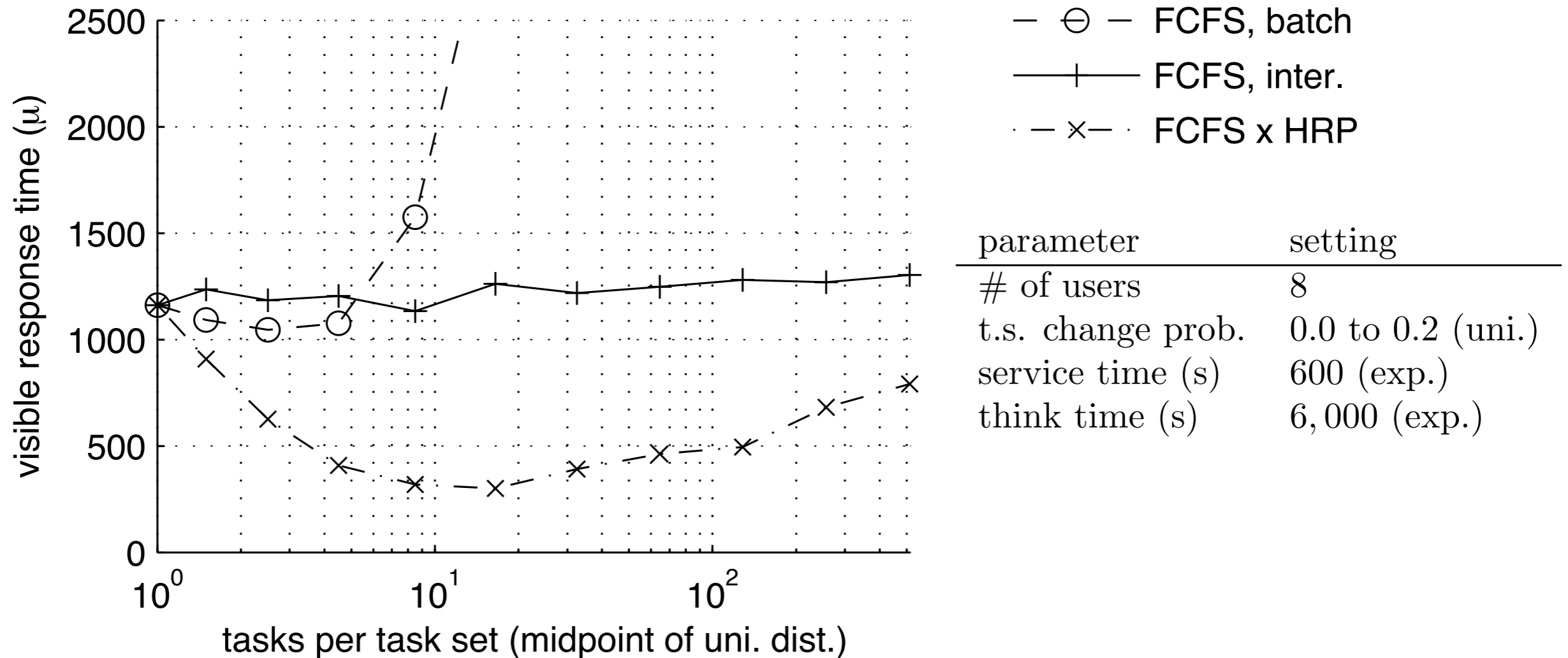
New disclosed queue policy



| parameter | setting |
|---------------------|----------------|
| # of users | 8 |
| # of tasks per t.s. | 1 to 15 (uni.) |
| service time (s) | 600 (exp.) |
| think time (s) | 6,000 (exp.) |

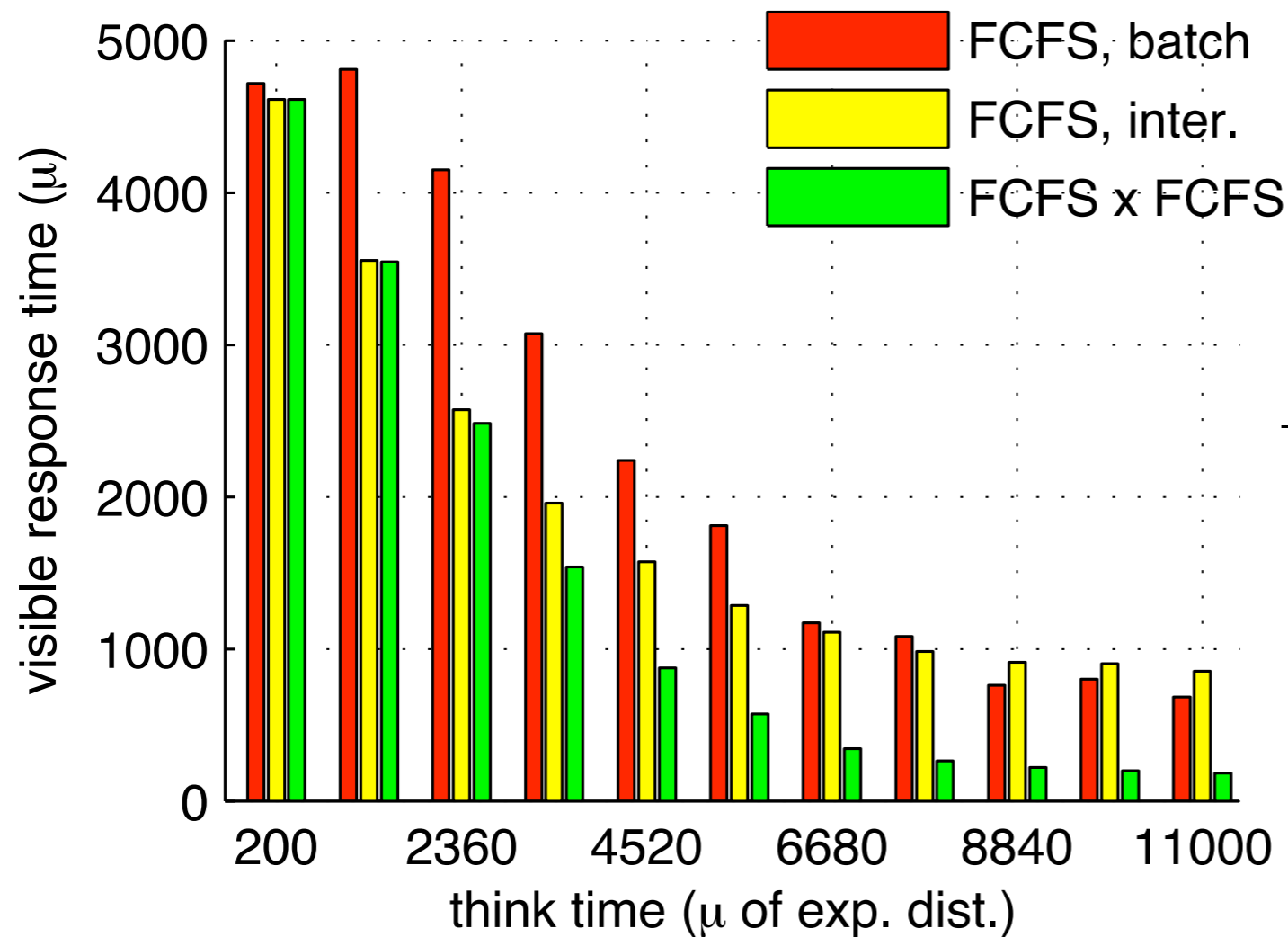
- HRP (highest request probability) better avoids executing unneeded speculation

Batchactive resilient to task set size



- Small task sets improve batchactive and batch; then **batch becomes unusable**

Think time helps batchactive more



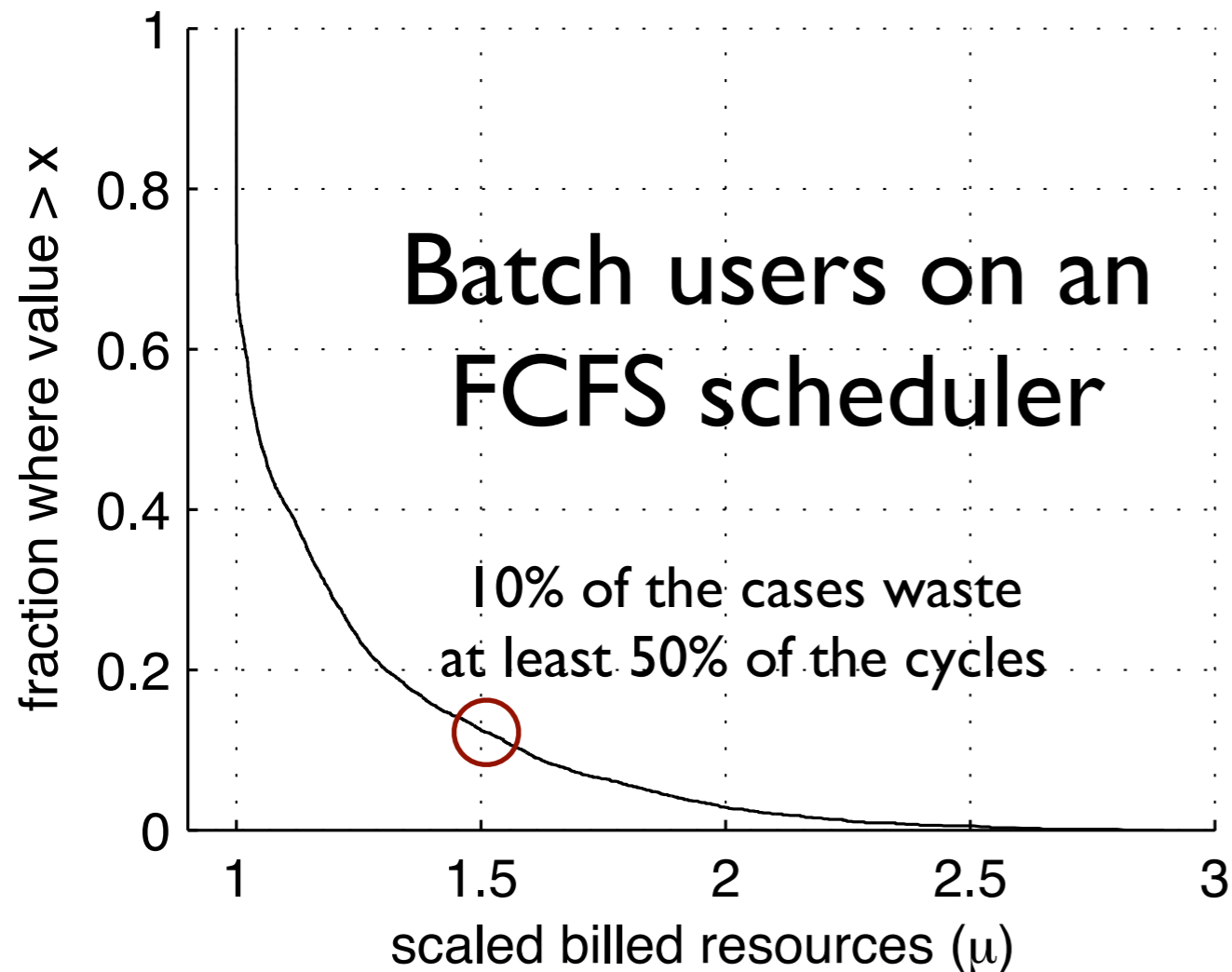
| parameter | setting |
|---------------------|-------------------|
| # of users | 8 |
| t.s. change prob. | 0.0 to 0.2 (uni.) |
| # of tasks per t.s. | 1 to 15 (uni.) |
| service time (s) | 600 (exp.) |

- Think time gives more opportunity to finish more pressing work earlier
- At limit, batch and batchactive converge, interactive significantly worse

Agenda

- Motivation & thesis in a nutshell
- Target applications
- Related work
- Batchactive scheduling
- Experimental design
- Think time
- Speculation
- ● Incentive pricing
- Proposed deployment
- Contributions & conclusions

Obstacle to speculation



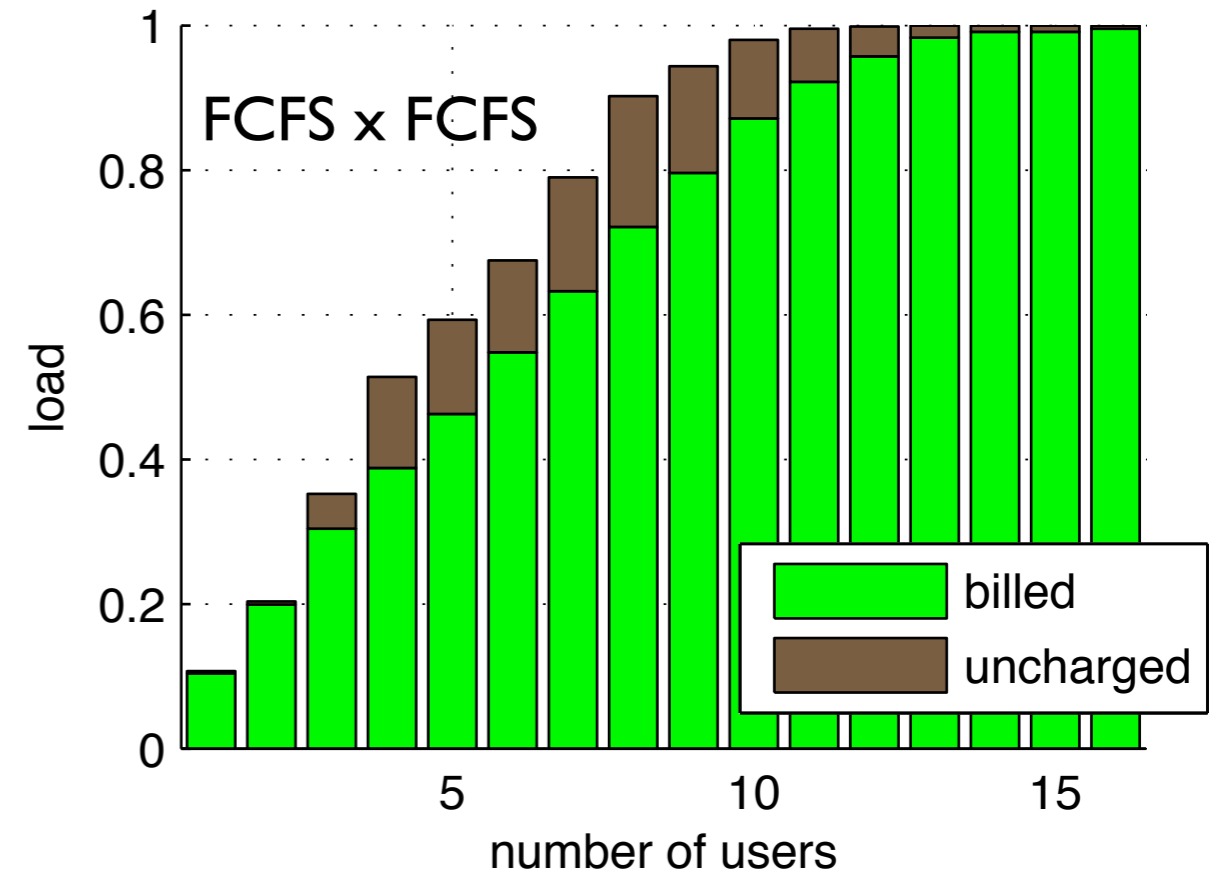
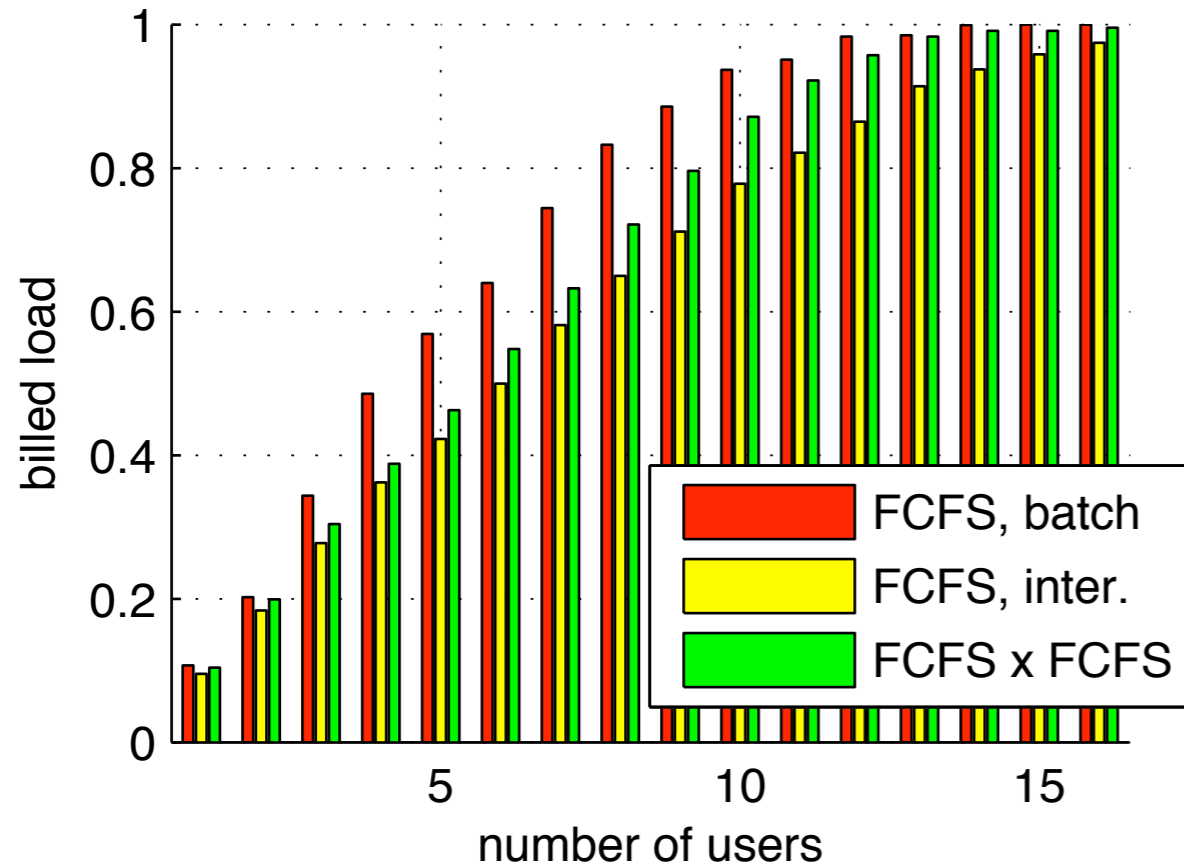
| param | range |
|---------------------|-----------------------|
| # of users | 1 to 16 |
| t.s. change prob. | 0.0 to 0.0–0.4 (uni.) |
| # of tasks per t.s. | 1 to 1–21 (uni.) |
| service time | 20 to 3,620 (exp.) |
| think time | 20 to 18,020 (exp.) |

- **Scaled billed resources:** billed divided by needed resources
- **Needless user charges**

Incentive pricing mechanism

- Users who submit speculative work traditionally risk paying for unneeded tasks
- To encourage deep speculation, batchactive systems **charge only for needed resources**
- **Batchactive users never pay for spec. tasks**
- Would a server use incentive pricing?
 - its revenue is the fraction of CPU time that is billed
 - seems to give something for free

Incentive pricing affects revenue



billed load is total load on traditional pricing, needed load on incentive pricing

| parameter | setting |
|---------------------|-------------------|
| t.s. change prob. | 0.0 to 0.2 (uni.) |
| # of tasks per t.s. | 1 to 15 (uni.) |
| service time (s) | 600 (exp.) |
| think time (s) | 6,000 (exp.) |

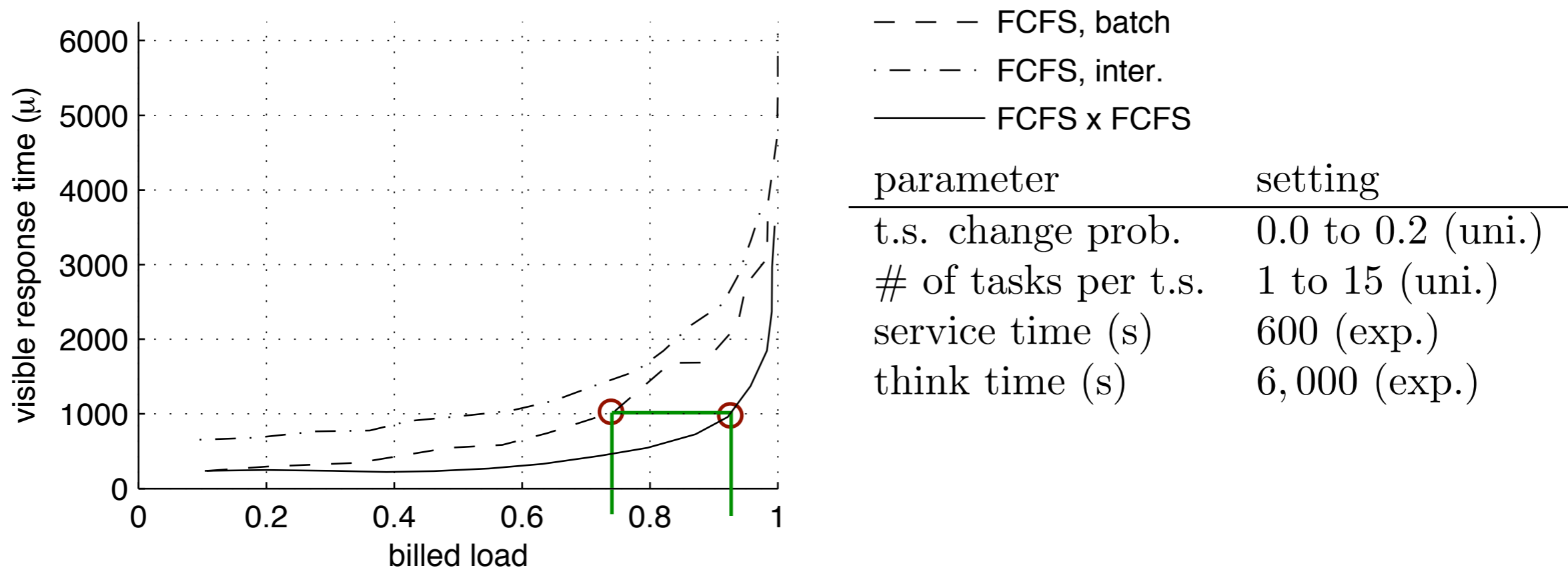
- Resource provider wishes to max. **billed load**
- Batch revenue highest; everything charged
- Inter. revenue lowest; its throughput is lowest

Argument for cost- and price-centers

- **Cost-center:** non-profit, IT center, university
 - make up equal cost by raising the price of requests
 - billing total is fixed, users happier
 - batch users will pay less; no spec. charges
 - interactive users will pay more, but receive better mean visible response time

Profit-center

- **Profit-center:** billing higher than cost
 - really no profit from batch users who would resist speculative charges
 - better revenue using trad. pricing results in poor delays for latency-sensitive users



- At any billed load, b.a. provides better mean visible response time
- Encourages additional users, deeper speculation, bigger tasks

Agenda

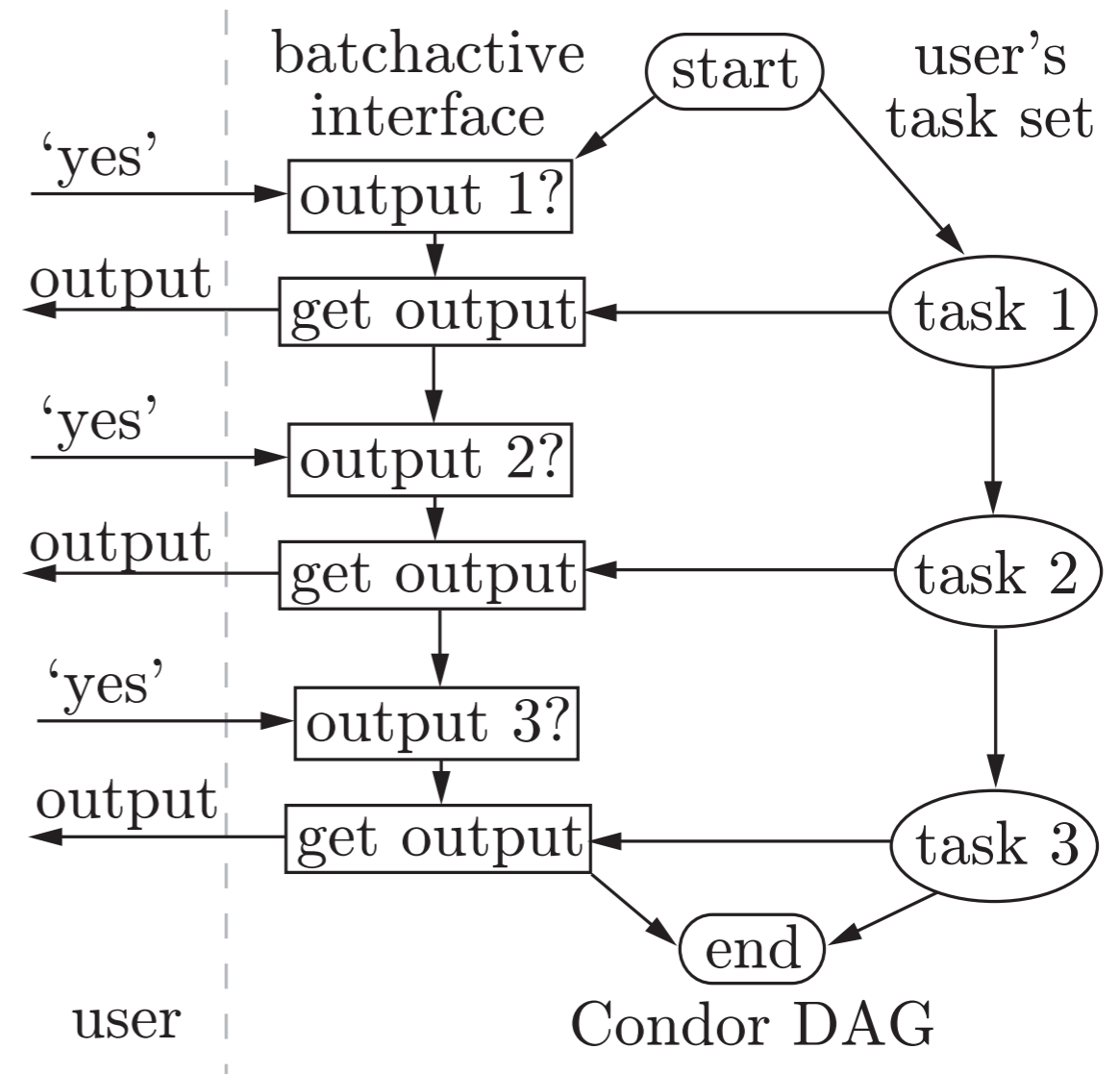
- Motivation & thesis in a nutshell
- Target applications
- Related work
- Batchactive scheduling
- Experimental design
- Think time
- Speculation
- Incentive pricing
- ● Proposed deployment
- Contributions & conclusions

Proposed deployment via Condor

- Condor: popular clustering system
 - distributes tasks to cluster nodes
 - hides details of remote exe., checkpointing
- Task submission uses **ClassAd** characteristics
 - per-user **Rank** specifies order
- **DAGMan** adds dependencies to execution
 - takes a directed acyclic graph as input
- Goal: support batchactive scheduling **without changing Condor** internals

Leveraging DAGMan & Rank

- **Batchactive interface**
 - offered through **DAGMan**
 - communicates candidates to central batchactive scheduler
- Scheduler modifies task **ranks** to approximate FCFS x FCFS, e.g.



Agenda

- Motivation & thesis in a nutshell
- Target applications
- Related work
- Batchactive scheduling
- Experimental design
- Think time
- Speculation
- Incentive pricing
- Proposed deployment
- Contributions & conclusions



Conclusions & contributions

- Identified important **speculative applications & user work patterns**
- *Non-obvious result:* **traditional scheduling non-ideal because think time is unexposed**
 - Advocate **visible response time**
- *Payoff:* Major benefits of **batchactive scheduling**
 - As good as best & **better in mid-high load**
 - Introduced a novel history-based policy, **HRP**
- *Obstacle removal:* **incentive pricing** encourages use
- Offer the highly-parameterizable **scheduling simulator** for download