



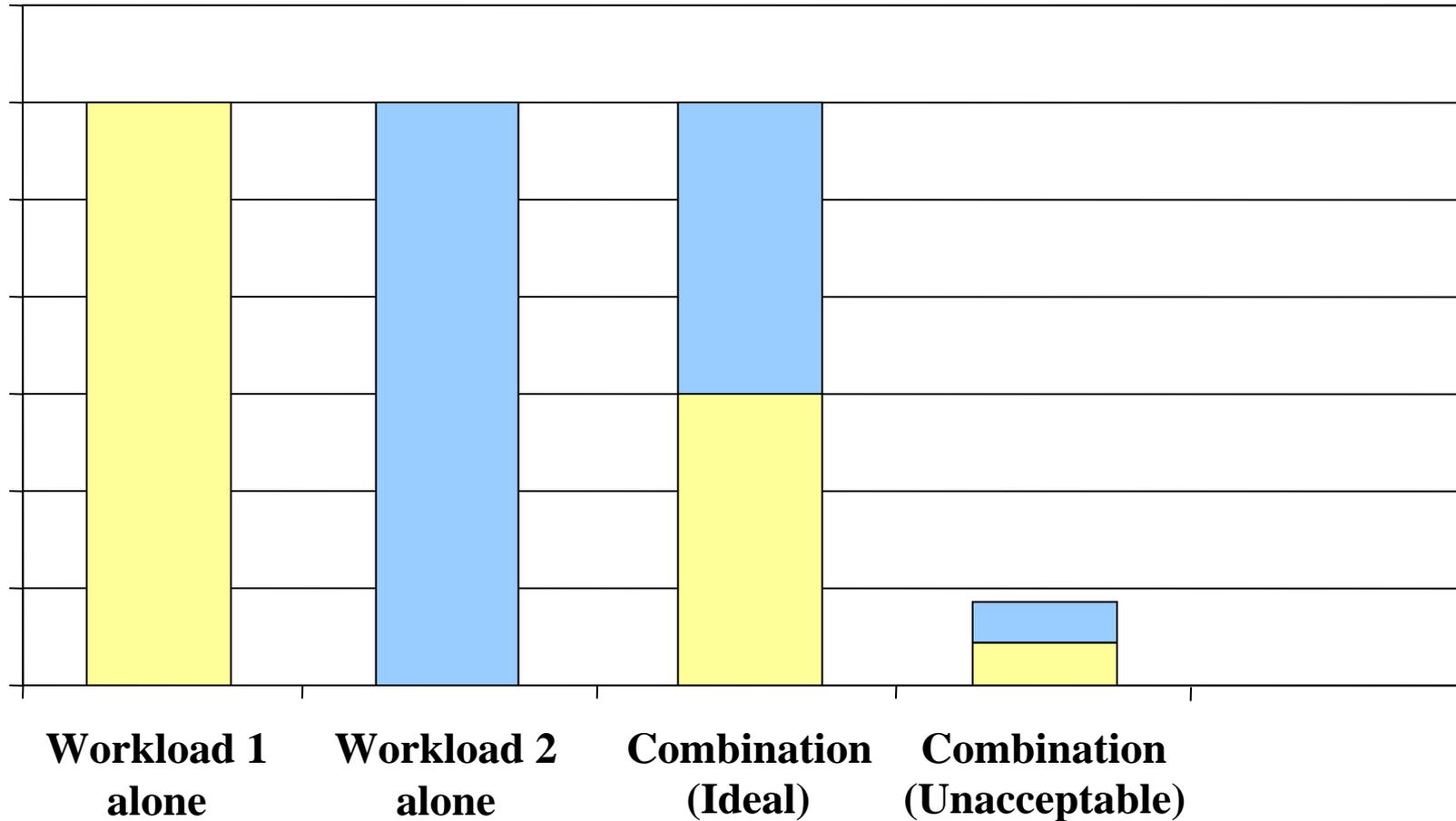
Performance insulation and predictability for shared cluster storage

Greg Ganger, Natassa Ailamaki, Garth Gibson

Whither shared storage clusters

- Contrasted to per-application/per-machine
 - sharing allows common namespace
 - sharing allows common provision+use of spare
 - including bursty usage
- But, interference can kill storage performance
 - Disk: “context switch” = mechanical seek (slow!)
 - Cache: what does time-sharing mean?
 - Cluster: coordinating timing across nodes

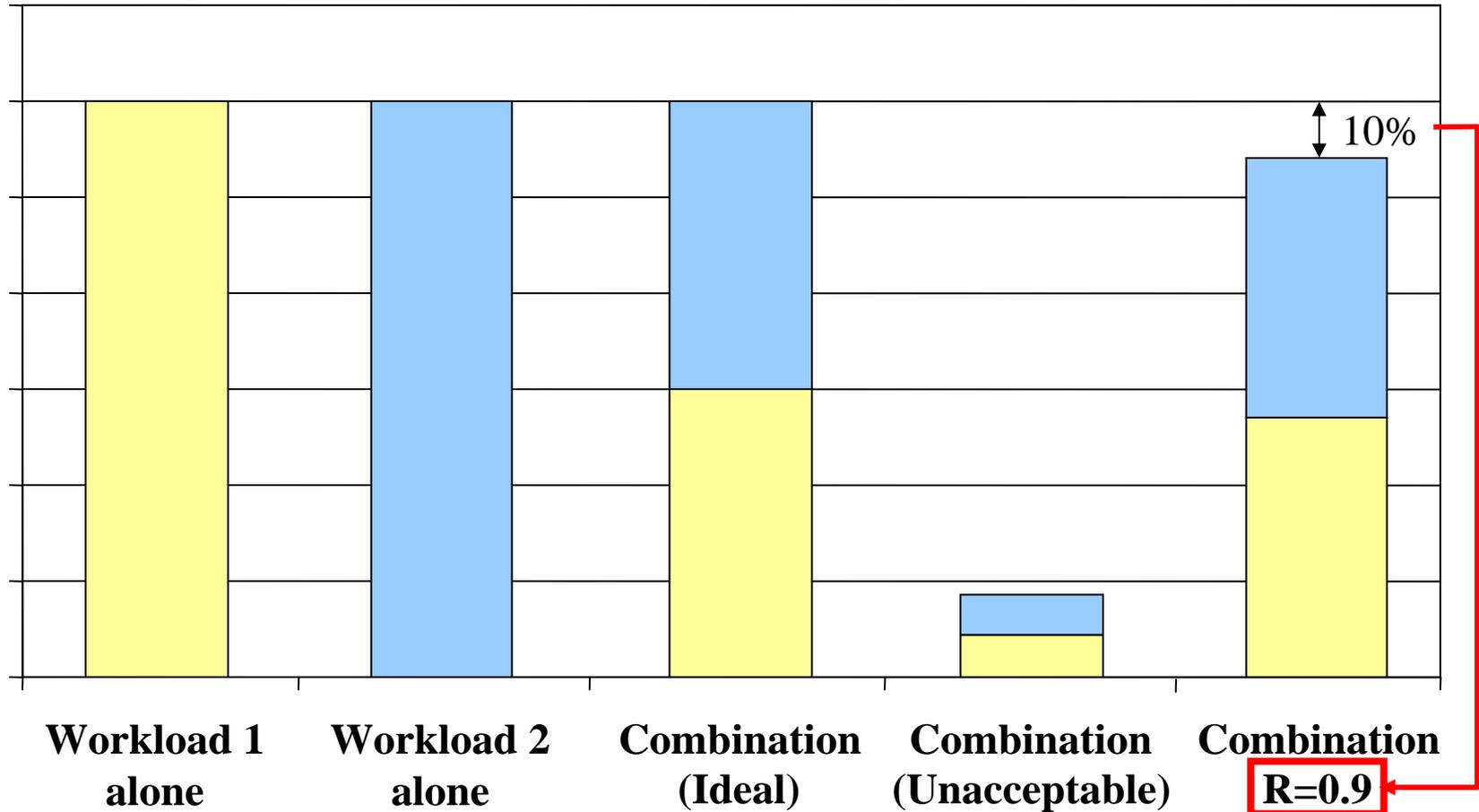
Example: sharing with two workloads



Predictability requires insulation

- Ideal: each of n workloads on a server
 - gets at least some explicit fraction of server “time”
 - e.g., $1/n$ or a chosen proportion
 - does not lose efficiency because of sharing
 - i.e., at least as efficient as when running alone
- Practical goal: an explicit “R-value” [Wachs07]
 - a configurable lower bound on efficiency
 - measured as throughput relative to non-sharing
 - adjusted according to the fraction of server time

Adding the R-value bars



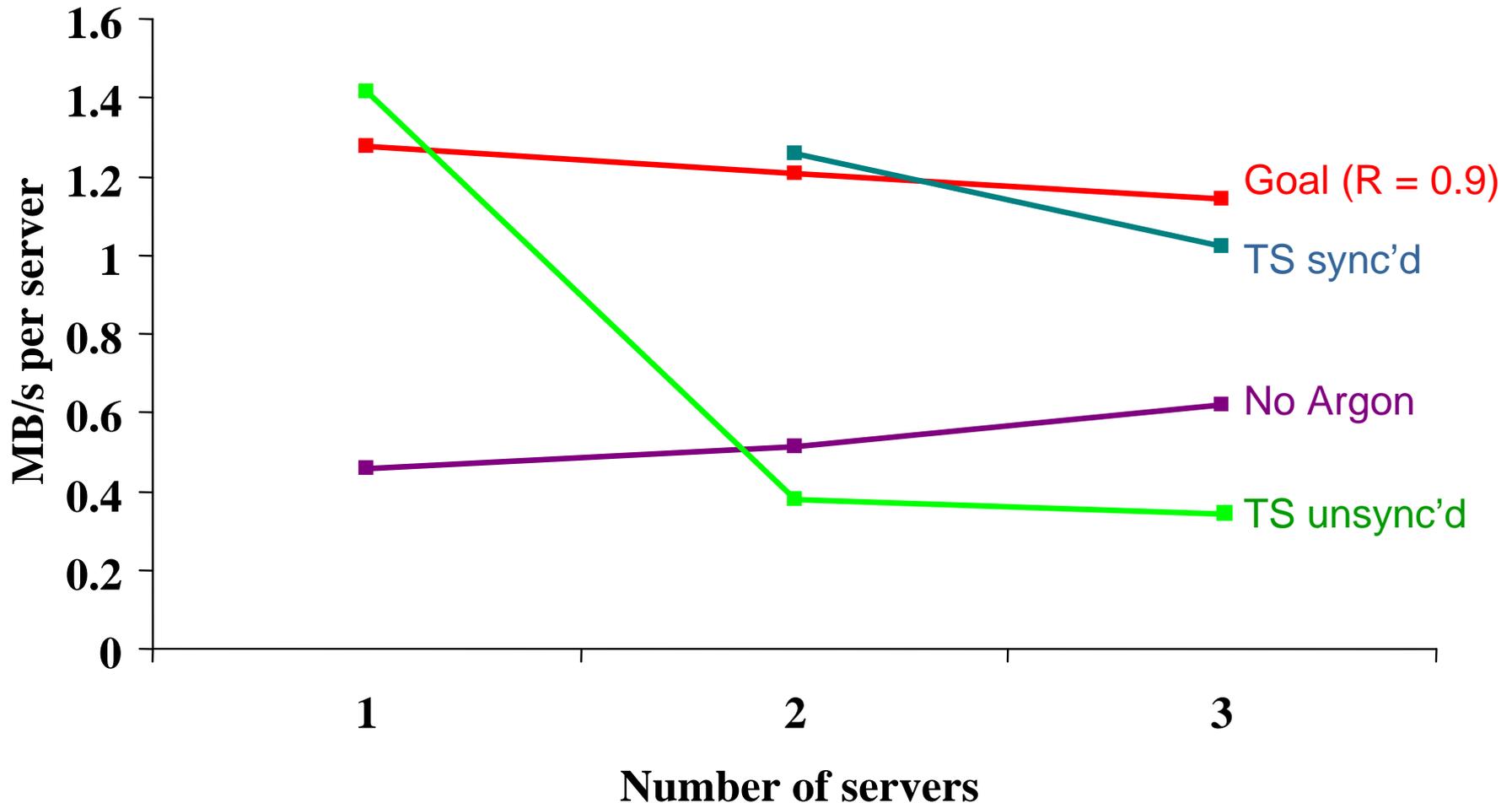
Early progress presented at FAST [Wachs07]

- Developed R-Value concept
- Identified single-server techniques required
 - a set of existing concepts can be meshed to do it
 - prefetching and write coalescing
 - cache partitioning
 - quanta-based timesharing of disk head
- Developed automated configuration policies
 - values computed based on chosen R-value

But, data will be striped over servers

- Data striped for performance (esp. bandwidth)
 - each client req. translates to multiple server accesses
 - client req. is “done” when all accesses are done
 - so, overall req. waits for the slowest one
 - unsynchronized quanta can lead to significant delays
 - so, need to coordinate quanta (a la synch spindles)

Focusing on one sharer's performance:



Must explicitly synchronize quanta

- Data striping means waiting for last server
 - and, subsequent requests wait for previous ones
- So, servers need to all be ready at same time
 - any given client gets full quanta of forward progress
- A combo of placement and scheduling
 - putting same number of quanta slots on each server
 - explicitly co-scheduling each workload's quanta
- ...and may need to de-synchronize net. xfers
 - see FAST 2008 paper on TCP Incast over Ethernet

Many add'l interesting problems

- Can R-value be used to help guide placement?
 - expose cache capacity insufficiency & latency impacts
 - expose particularly complementary combinations (?)
- When can we both insulate and exploit slack?
 - merging quanta when it's ok
- What should the QoS control system look like?
 - insulation has a role (provides predictable bounds)
 - explicit feedback on placement has a role
 - safe, insulated use of slack has a role
- What happens when workloads change?
 - need to determine timescale used to define
 - need periodic re-analysis to identify new interference

Project participants

- Matthew Wachs, 3rd year Ph.D. student
- Elie Krevat, 1st year Ph.D. student
- Mike Abd-El-Malek, 4th year Ph.D. student
- Chuck Cranor, system scientist
- Spencer Whitman & Manish Prasad, research programmers
- Greg, Natassa, Garth

- Graduated students:
 - PhDs: Eno Thereska (Microsoft), Mike Mesnier (Intel)
 - MS: Deepti Chheda, Mikhail Chianani, Chandramouli Rangarajan, Aditya Sethuraman

Ending

- End goal: predictable (w/QoS) shared storage
 - bounded efficiency loss to interference (R-value)
 - configurable fractions of server time per “application”
- Need mechanisms at multiple levels
 - cache and disk time allocation to achieve R-value
 - multi-server coordination to avoid delays and incast
 - QoS control to manage resource priorities and slack
- Solid results on 1st item and progress on 2nd

For more information:
<http://www.pdl.cmu.edu/>

Greg.Ganger@cmu.edu
Director, Parallel Data Lab

