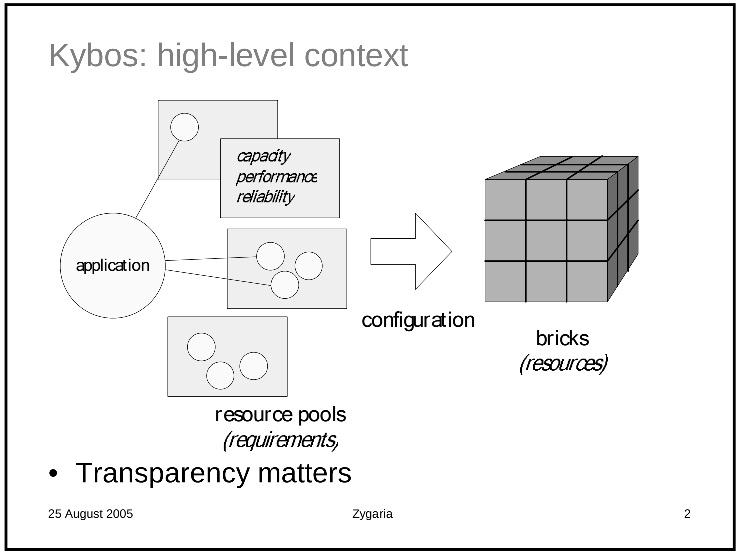
Zygaria: storage performance as a managed resource

Richard Golding, Theodore Wong, Caixue Lin, and Ralph Becker-Szendy

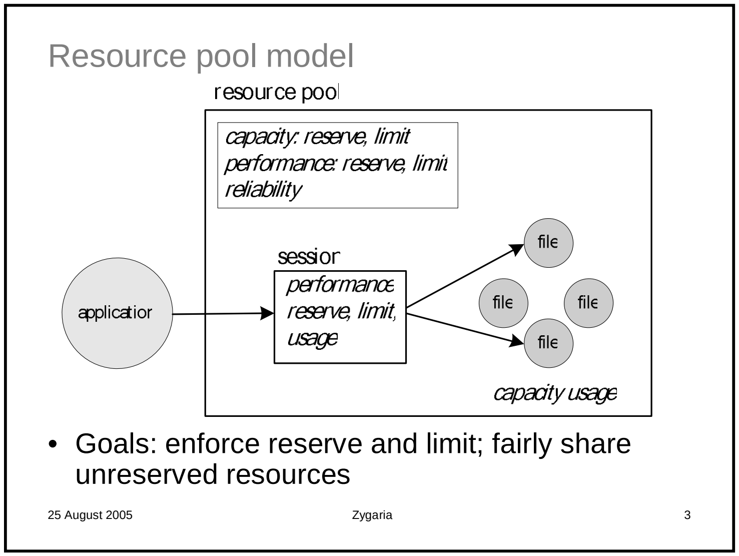
IBM Almaden Research Center

25 August 2005



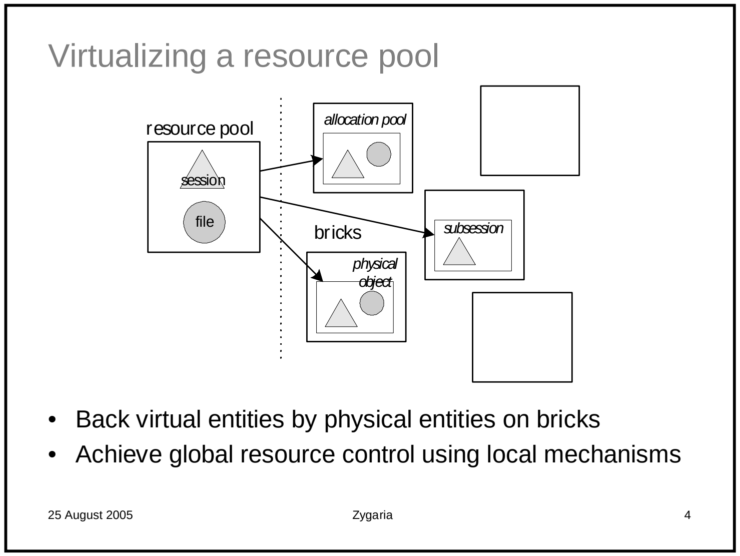
Kybos: high-level context

- Transparency matters



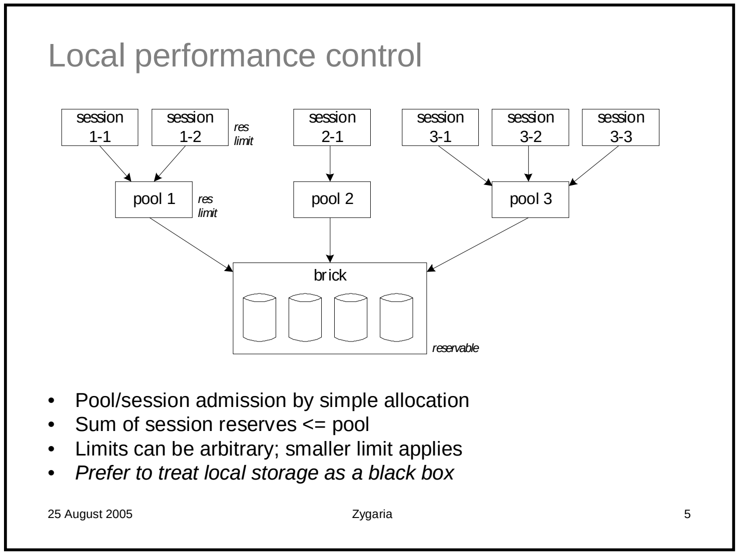
Resource pool model

- Goals: enforce reserve and limit; fairly share unreserved resources



Virtualizing a resource pool

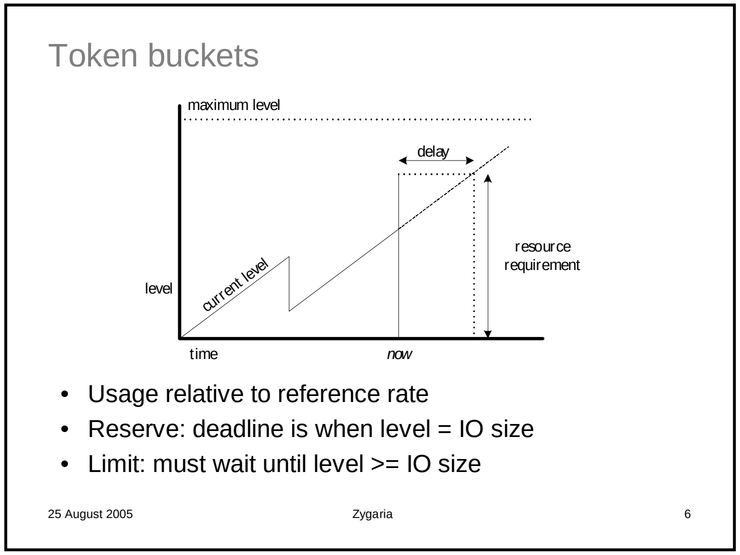
- Back virtual entities by physical entities on bricks
- Achieve global resource control using local mechanisms



Local performance control

- Pool/session admission by simple allocation
- Sum of session reserves <= pool
- Limits can be arbitrary; smaller limit applies
- *Prefer to treat local storage as a black box*

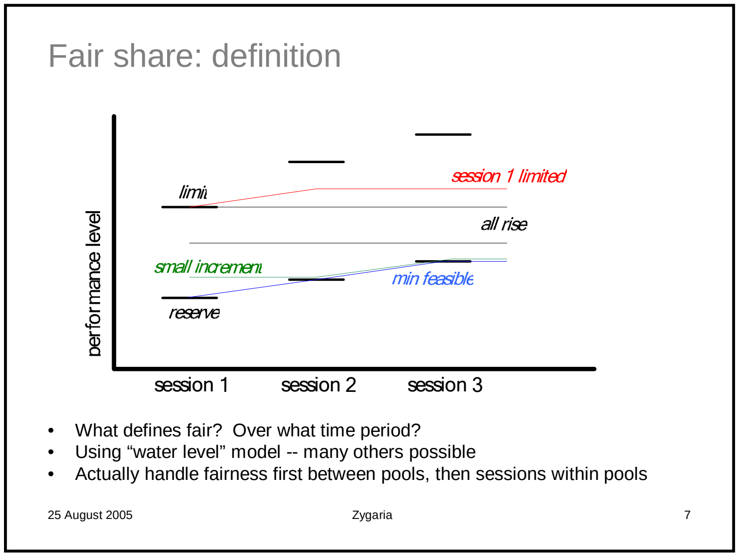
25 August 2005 Zygaria 5



Token buckets

- Usage relative to reference rate
- Reserve: deadline is when level = IO size
- Limit: must wait until level >= IO size

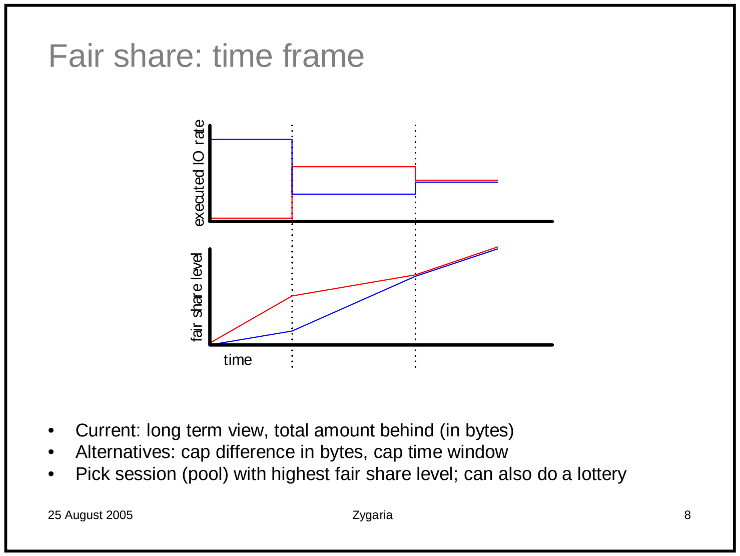
25 August 2005 Zygaria 6



Fair share: definition

- What defines fair? Over what time period?
- Using “water level” model -- many others possible
- Actually handle fairness first between pools, then sessions within pools

25 August 2005 Zygaria 7

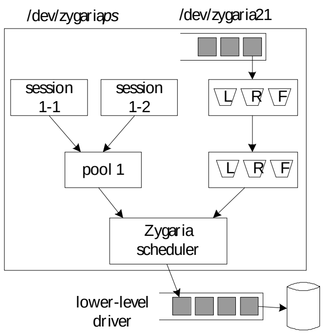


Fair share: time frame

- Current: long term view, total amount behind (in bytes)
- Alternatives: cap difference in bytes, cap time window
- Pick session (pool) with highest fair share level; can also do a lottery

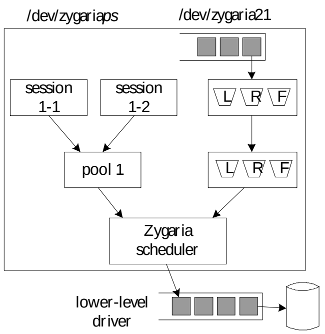
25 August 2005 Zygaria 8

Zygaria: algorithm



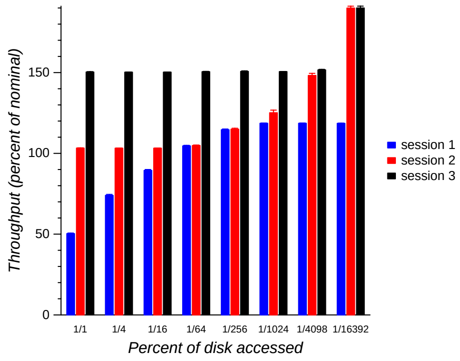
- Token buckets + EDF + slack stealing
- Compute IO *release times*
  - Later of session and pool limit
- Then compute IO *deadlines*
  - If in past, send now
  - Pool past deadline => send earliest session
- If slack and lower-level queue < *q* IOs:
  - Pick pool with highest fair share
  - Pick highest session in that pool
- Repeat until nothing more to do
- Trigger on IO arrival, completion, deadline timer

Zygaria: implementation

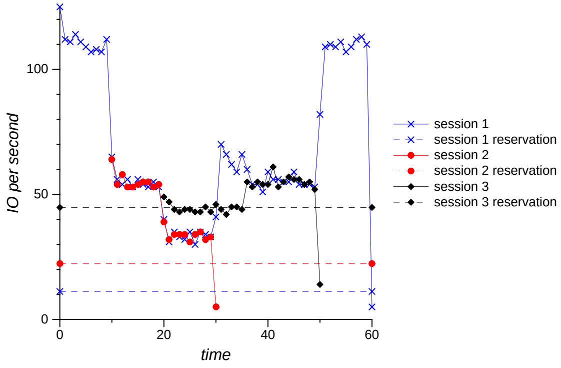


- Loadable driver, Linux 2.6.11
- 2116 lines of C
- *q*=10 outstanding IOs
- Standard Linux IDE driver
- 1.2 GHz Pentium III
- 120 GB, 7200 rpm, IDE

Basic results: sharing



Basic results: over time



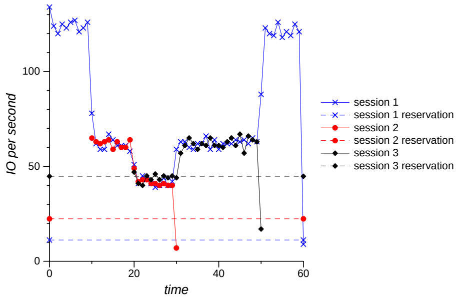
- Disk can do 112 uniformly-distributed random IOps
- All sessions use closed/20 outstanding/0 think time generators, random 1KB requests uniform over whole disk

Overhead

Configuration	Total CPU (%)	Throughput (IOPS)
No Zygaria	1.11	145
1 pool, 1 sess, *q*=10	1.07	112
10 pool, 10 sess, *q*=10	1.24	112
1 pool, 1 sess, *q*=100	0.88	145
10 pool, 10 sess, *q*=100	0.78	145

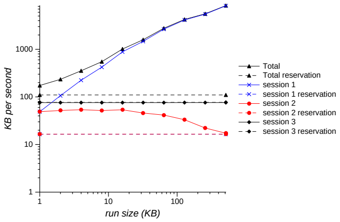
- CPU overhead is in the noise
- Expect proportional to number of active sessions/pools
- Throughput not so good – fewer IOs for head scheduling

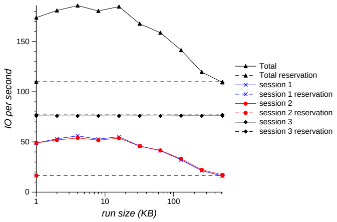
Adding batching



- Send IOs in batches of up to 10 from sessions
- More aggressive about slack: adapt *q* based on reservation of active sessions, observed disk performance
- Higher total throughput
- Retain performance for closed/1 workloads

Combining bandwidth and IO rate

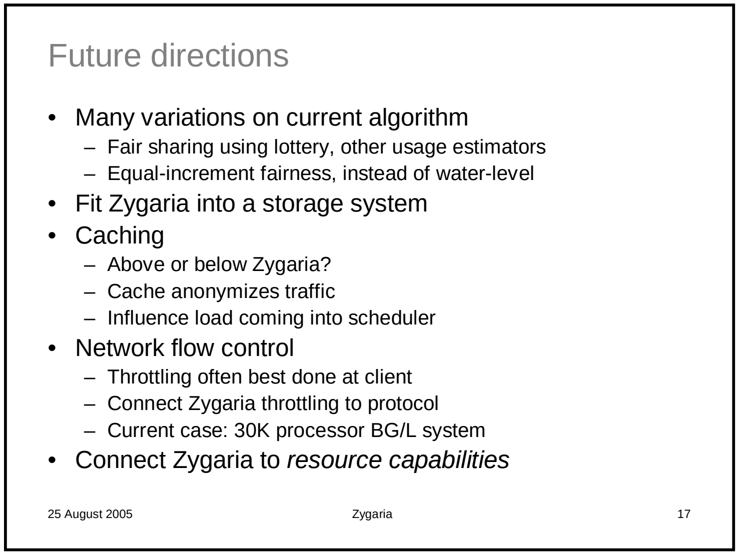




- Use IO *runs* per second; run is up to *r* KB sequential requests
  - Less bias toward random traffic, handle multiple IO sizes
  - Just group IO requests in a session's queue
- Parameter *r* manually set; needs to be global for comparability between systems

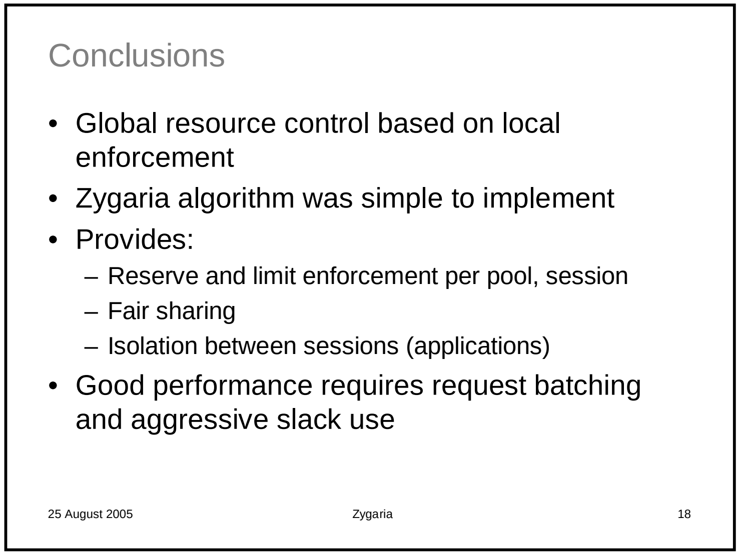
Related work

- IO and soft realtime scheduling
  - Media-oriented IO scheduling: Clockwise, Cello
  - Zygaria provides looser scheduling than traditional SRT
- Hierarchical resource allocation
  - UCSC hierarchical disk sharing, Q-RAM, HLS, DQM
- Fair share scheduling
  - Lottery scheduling, YFQ
- Façade, SLEDrunner
  - Adaptive mechanisms, focused on latency
  - Built around an EDF scheduler
  - Zygaria provides stronger guarantees, better control transparency



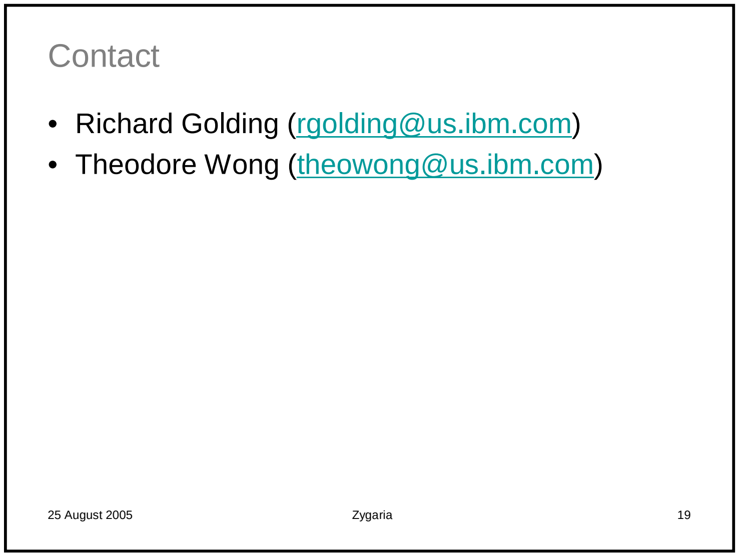
Future directions

- Many variations on current algorithm
  - Fair sharing using lottery, other usage estimators
  - Equal-increment fairness, instead of water-level
- Fit Zygaria into a storage system
- Caching
  - Above or below Zygaria?
  - Cache anonymizes traffic
  - Influence load coming into scheduler
- Network flow control
  - Throttling often best done at client
  - Connect Zygaria throttling to protocol
  - Current case: 30K processor BG/L system
- Connect Zygaria to *resource capabilities*



Conclusions

- Global resource control based on local enforcement
- Zygaria algorithm was simple to implement
- Provides:
  - Reserve and limit enforcement per pool, session
  - Fair sharing
  - Isolation between sessions (applications)
- Good performance requires request batching and aggressive slack use



Contact

- Richard Golding (rgolding@us.ibm.com)
- Theodore Wong (theowong@us.ibm.com)