sinfonia: a new paradigm for building scalable distributed systems

marcos k. aguilera hp labs
arif merchant hp labs
mehul shah hp labs
alistair veitch hp labs
christos karamanolis vmware

motivation

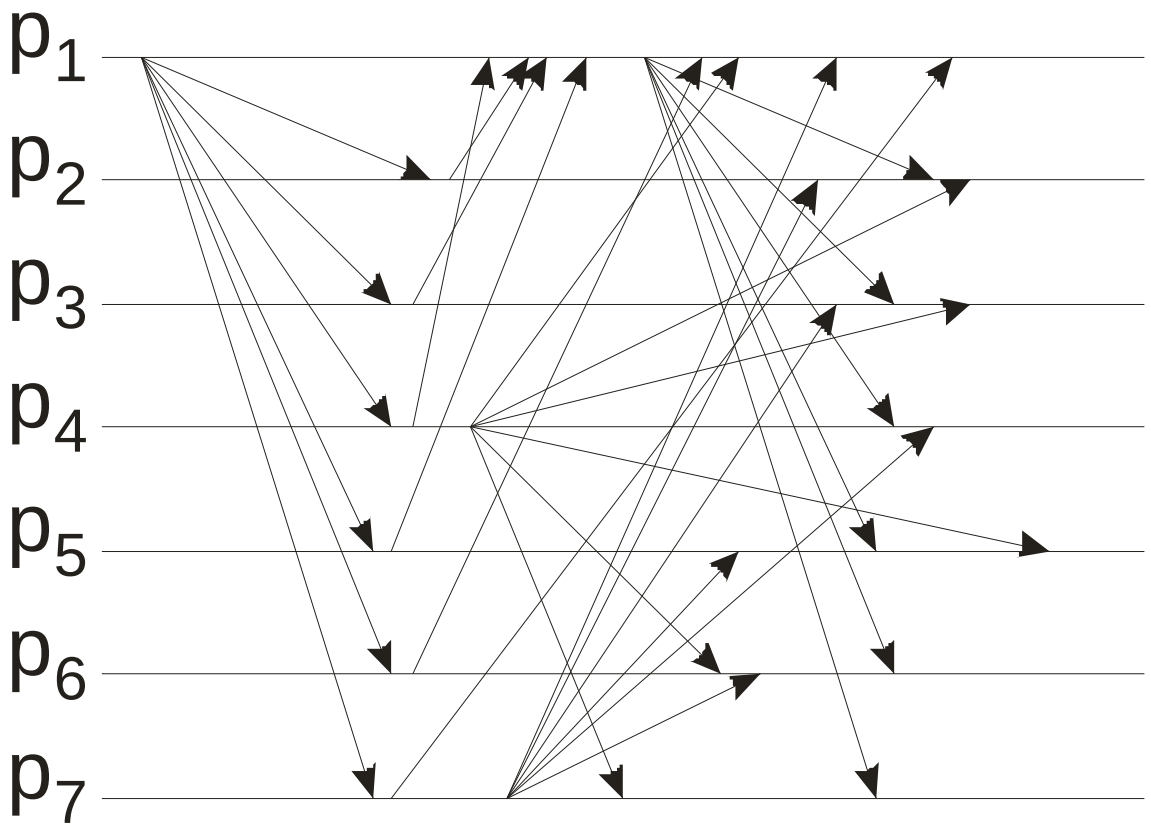
corporate data centers are growing quickly

- companies building large data centers
- 10000s servers and more
- businesses want to serve the world

need distributed applications that scale well

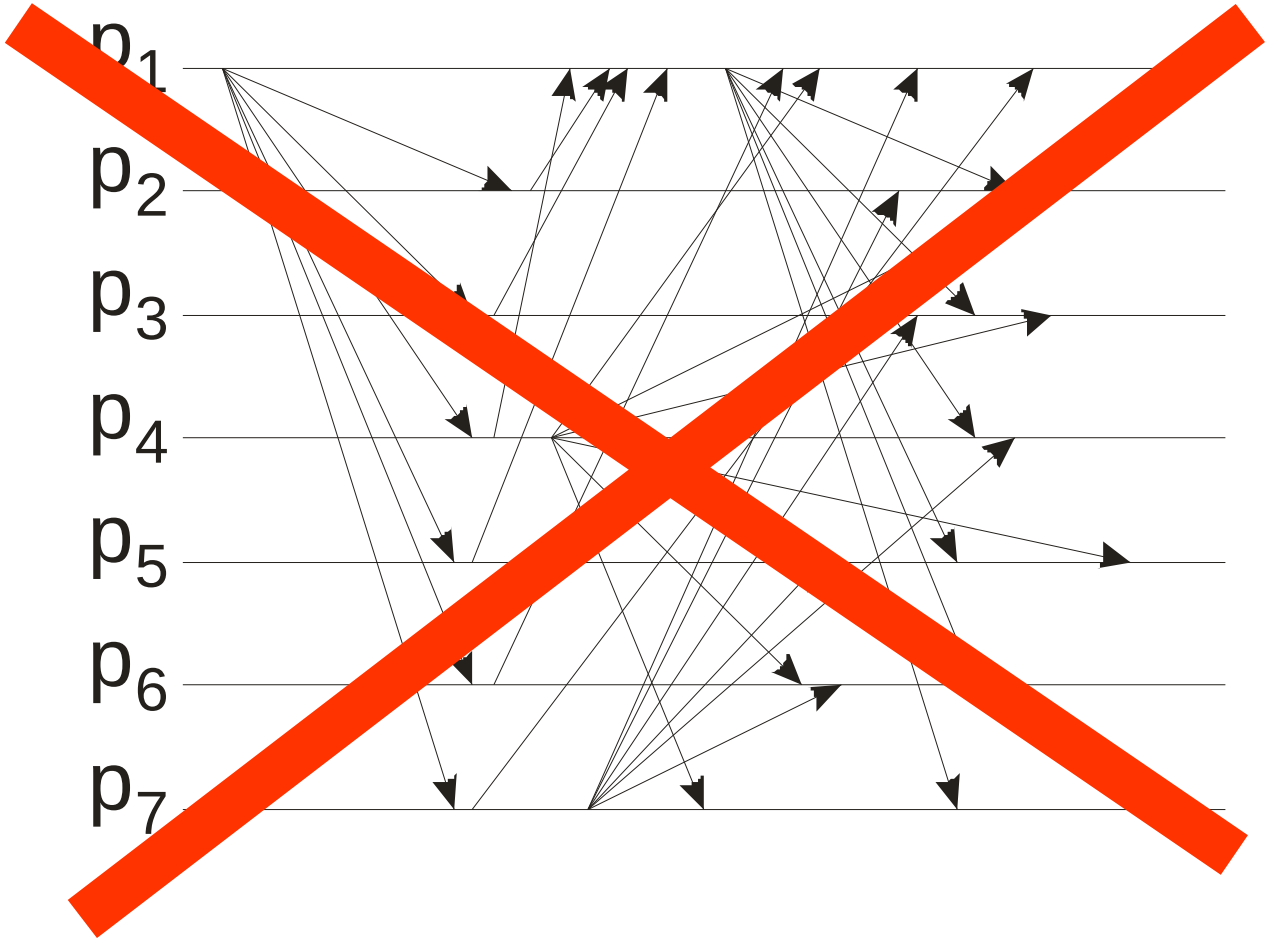
hp invent

current distributed applications often involve complex protocols



hp invent

wouldn't it be nice to avoid such protocols?



hp invent

focus

systems within a data center

- network latencies usually small and predictable
- nodes may crash, sometimes all of them
- stable storage may crash too

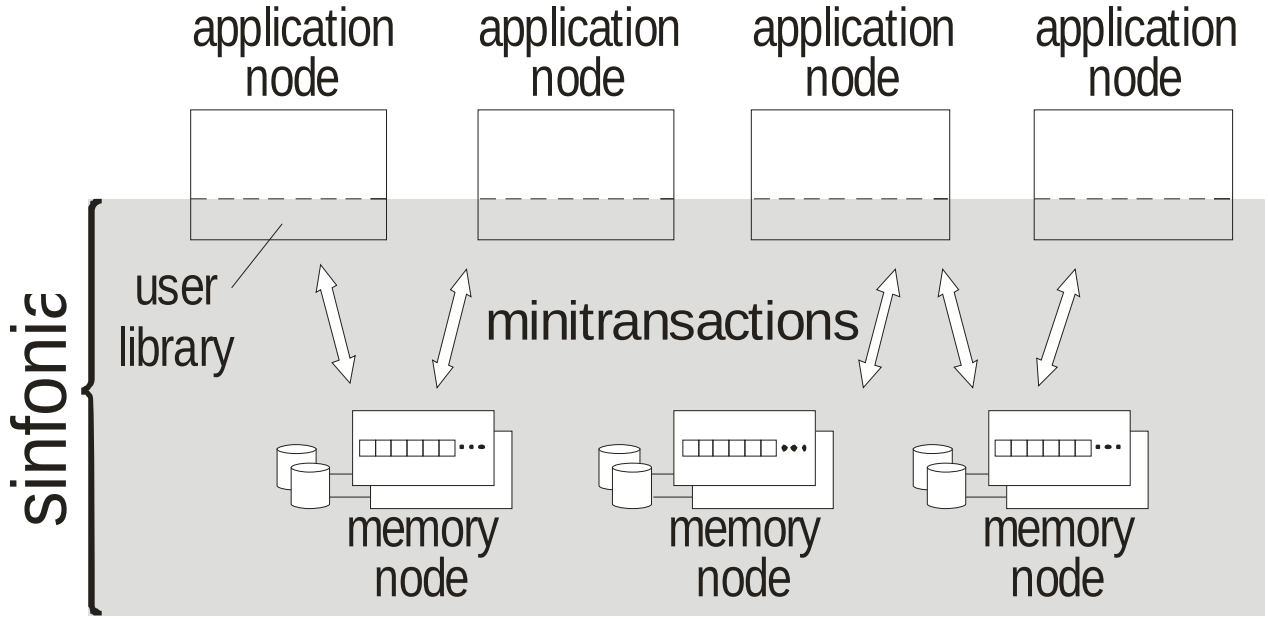
infrastructure applications

- applications that support other applications
- reliable, fault-tolerant, consistent
- examples: cluster file systems, distributed lock managers, group communication services, distributed name services

hp invent

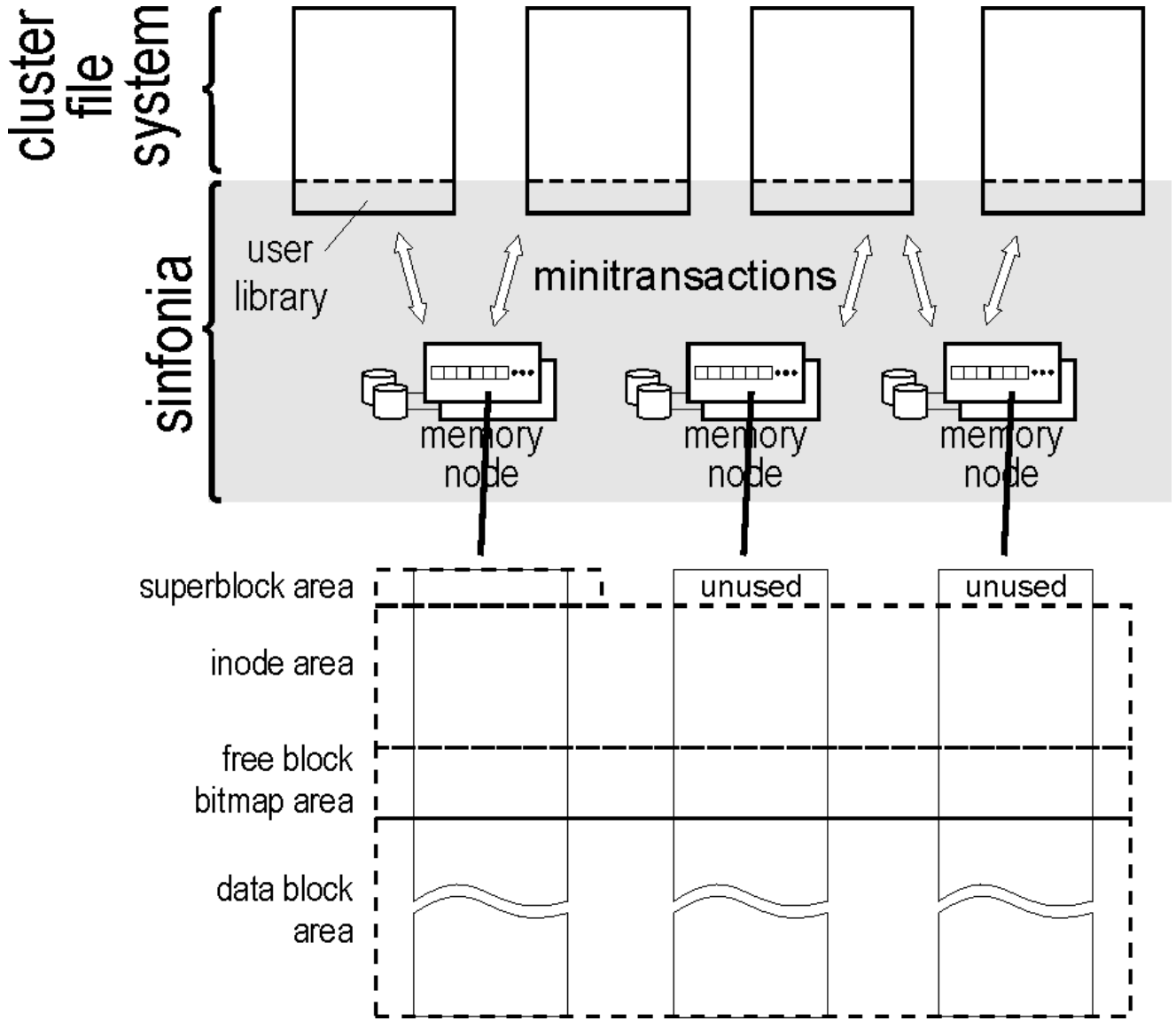
our approach
to developing distributed applications

- developers use sinfonia, a data sharing service
  - data stored in memory nodes, each exporting a linear address space
  - no structure on data imposed by sinfonia
  - streamlined minitransactions
- transform problem of protocol design into easier problem of shared data structure design



hp invent

example application: cluster file system
details later

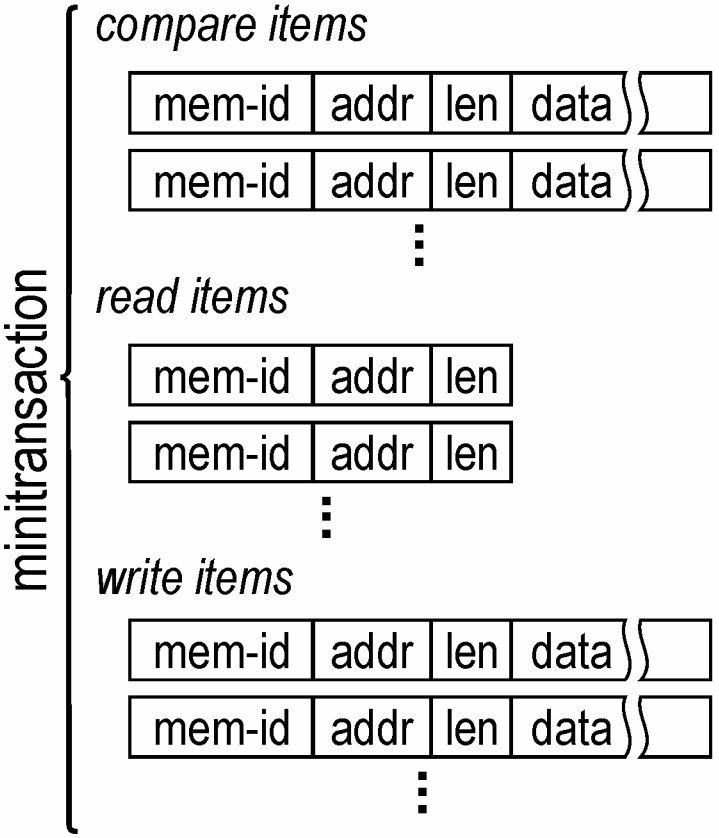


sinfonia minitransactions

- operate on data at memory nodes
- provide ACID properties
  - atomicity, consistency, isolation, durability
- designed to balance power and efficiency
- efficiency
  - few network roundtrips to execute
- power
  - flexible, general-purpose, easy to understand and use
- result
  - a lightweight, short-lived type of transaction over unstructured data

hp invent

minitransaction in detail



semantics

- check data indicated by *compare items* (equality comparison)
- if all match then
  retrieve data indicated by *read items*
  modify data indicated by *write items*

example

```
t = new Minitransaction;
t->cmp(hostX, addrX, len, 1);
t->write(hostY, addrY, len, 2);
t->write(hostZ, addrZ, len, 2);
status = t->exec_and_commit();
```

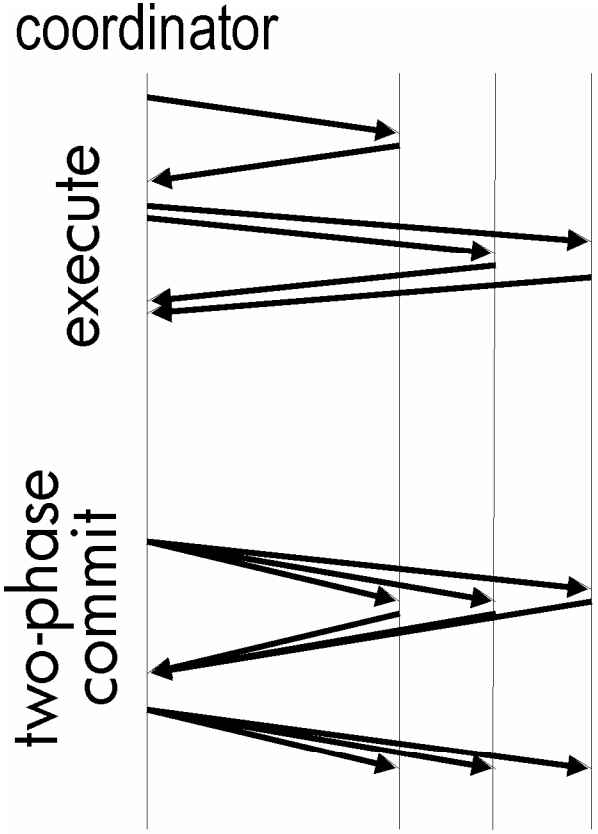
power of minitransactions

examples of what one minitransaction can do

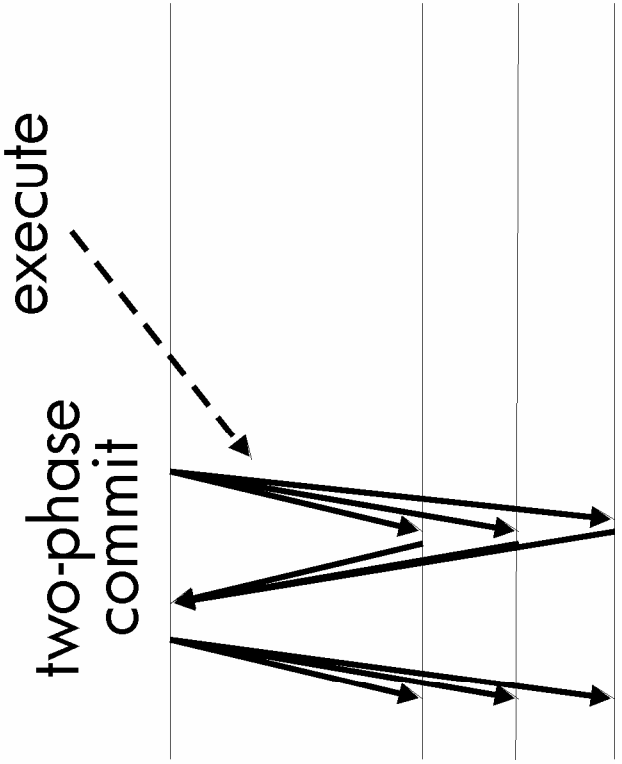
1. atomic swap operation
2. atomic read of many data
3. try to acquire a lease
4. try to acquire multiple leases atomically
5. change data if lease is held
6. validate cache then change data
   (e.g., optimistic concurrency control)

hp invent

minitransaction efficiency:
piggybacking execution onto two-phase commit



traditional transactions



sinfonia minitransactions

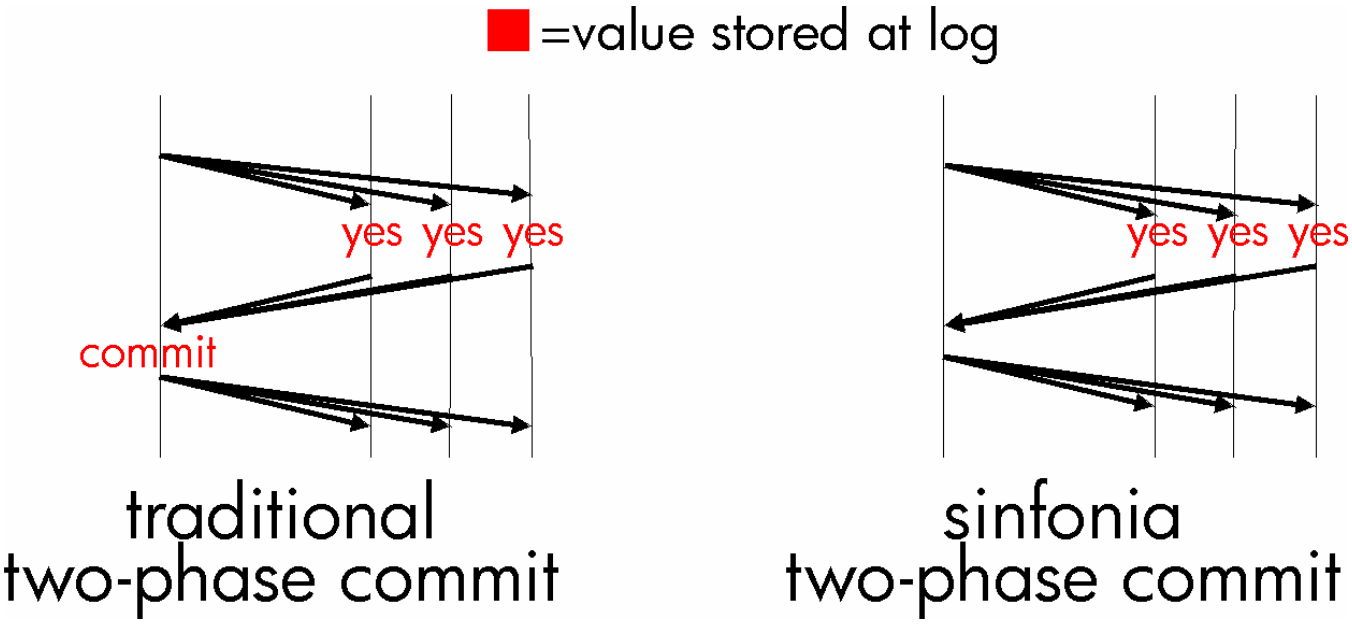
minitransaction efficiency:
running at application node

- commit coordinator runs at application node
  to save a network roundtrip
- problem: coordinator may crash and not recover
  application node outside of sinfonia control
- cannot block transactions forever in this case
- 3-phase commit expensive
- solution: a new two-phase commit protocol

hp invent

new two-phase commit

- transaction committed iff
  all participant memory nodes log “yes” vote
  compare: transaction committed iff coordinator logs “commit” decision
- but: transaction blocks while memory node is crashed
- recovery and garbage collection more involved (see paper)



hp invent

other features of sinfonia
not covered in detail in this talk

configurable fault tolerance

cost, performance, resiliency trade offs

memory node replication

can have mirrors of memory nodes for better availability

transactional backups

way to capture a transactionally consistent full image

debugging facilities

transaction log (gc disabled) to review past

hp invent

using sinfonia: applications

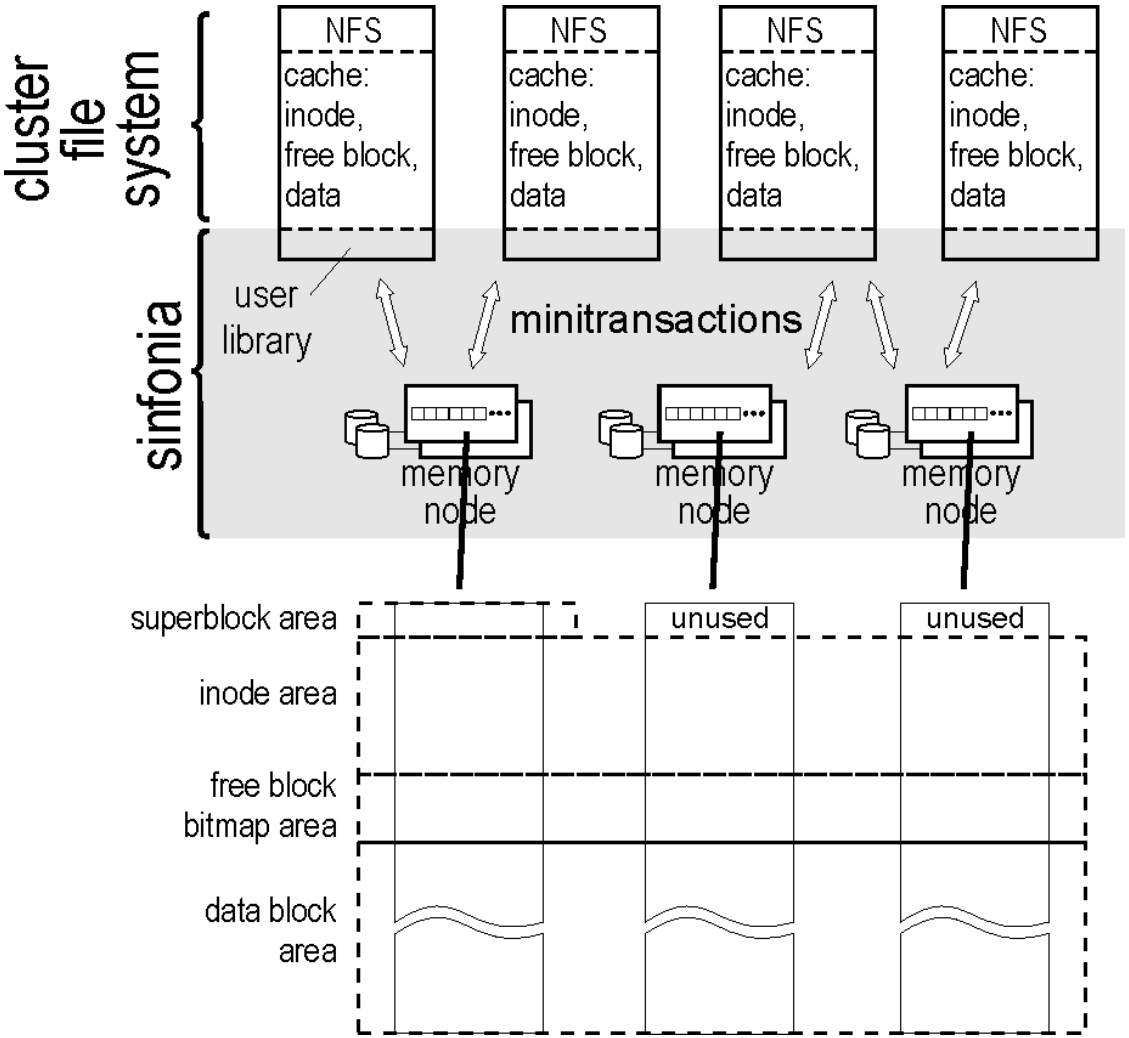
two complex distributed applications

1. sinfoniaFS: cluster file system
   - hosts share the same set of files, files stored in sinfonia
   - fault tolerant
   - scalable: performance improves with more memory nodes

2. sinfoniaGCS:
   group communication service [birman, joseph 1987]
   - “chat room” for distributed applications
   - nodes can join or leave the room, notifications of who joins/leaves
   - nodes can broadcast messages to room, messages totally ordered

hp invent

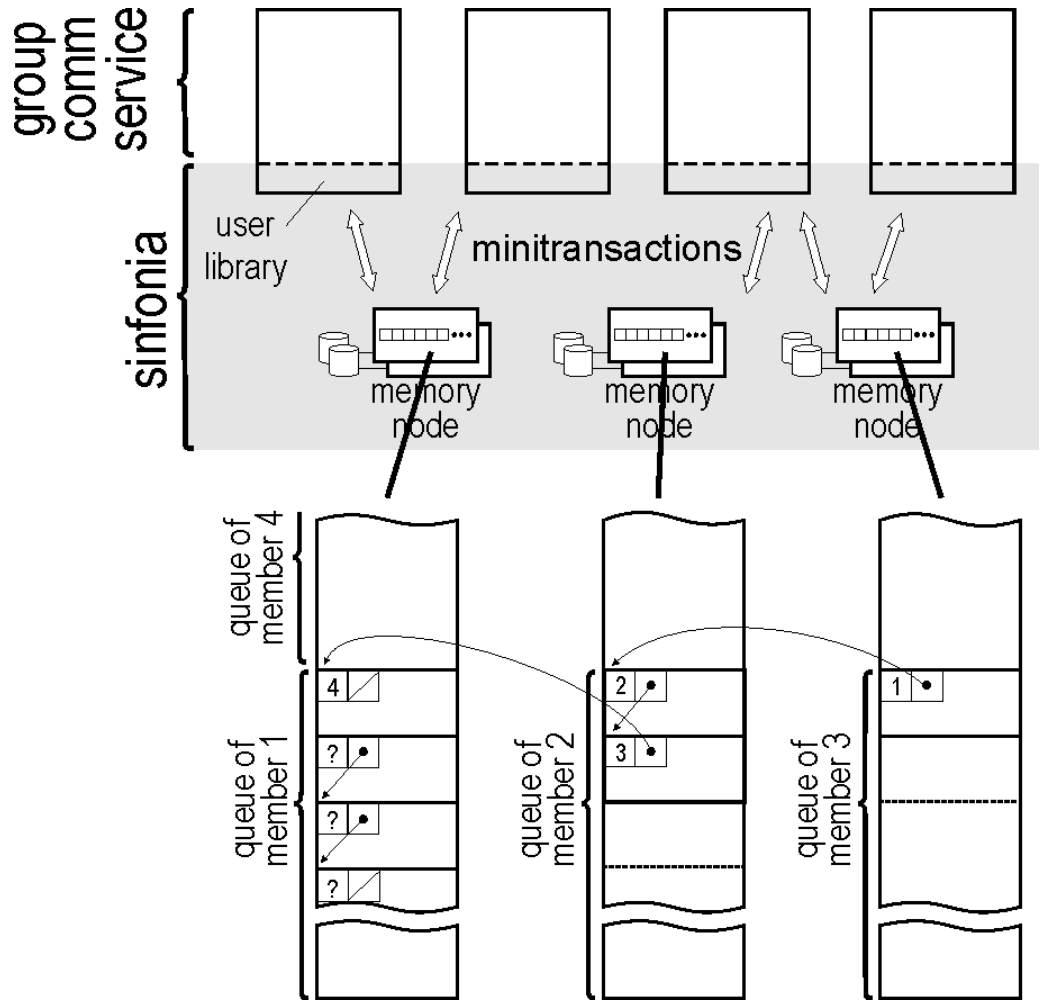
1. sinfoniaFS design



- exports NFS interface
- each NFS operation: one minitransaction
- general template:
  - validate cache (cmp items)
  - modify data (write items)

hp invent

2. sinfoniaGCS design



- each member has private queue in sinfonia
- broadcast msg:
  1. copy msg to queue
  2. thread msg in global order
- join or leave:
  1. acquire lease
  2. update member list
  3. release lease

hp invent

evaluation

sinfonia service

- scalability
- performance under contention
- ease of use
- *paper: benefits of streamlined minitransactions* (1.4x to 11.1x throughput improvement)

cluster file system application

- performance and scalability

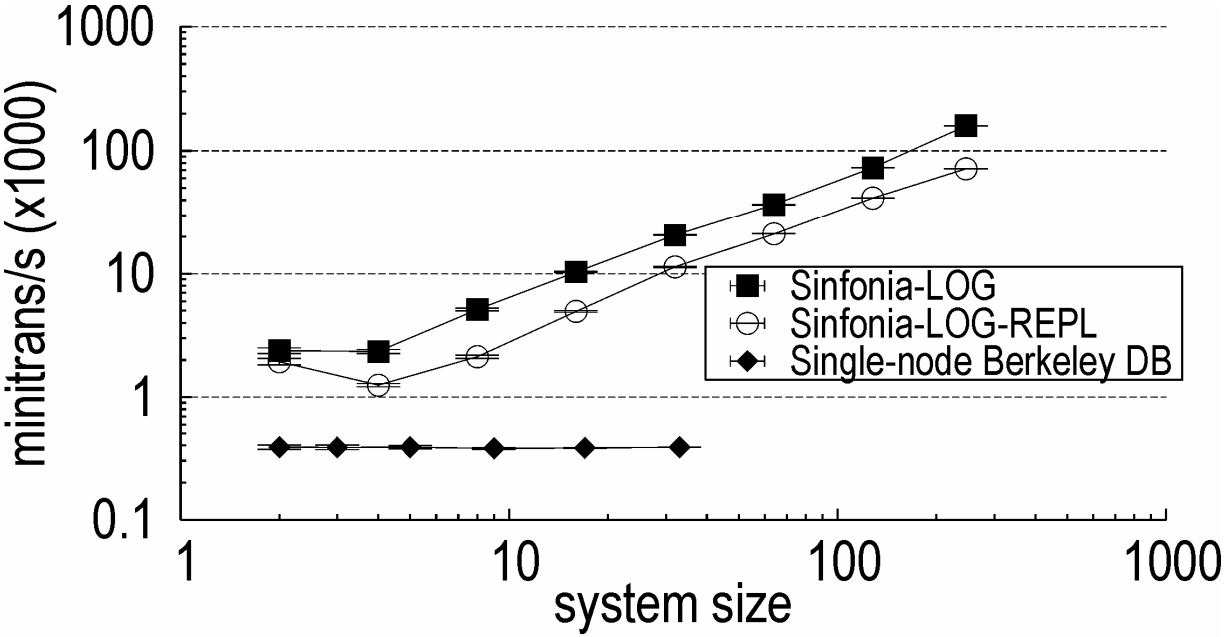
group communication application

- performance and scalability

hp invent

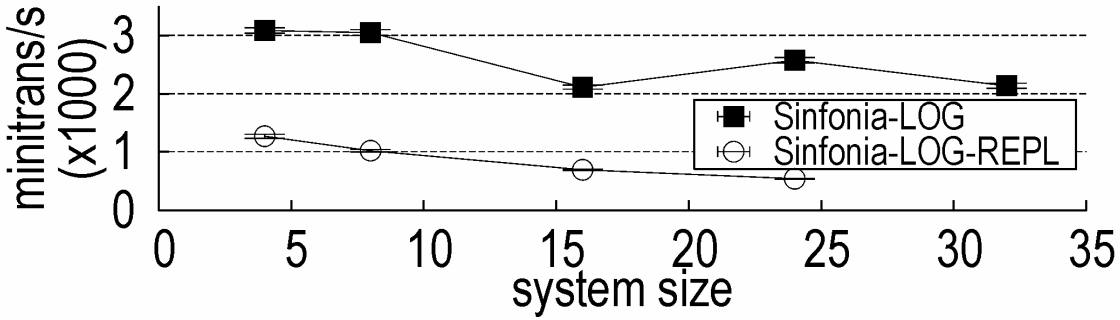
sinfonia service: scalability

minitransaction spread: number of memory nodes in a minitransaction



minitransaction spread=2:

usually within 85% of ideal scalability



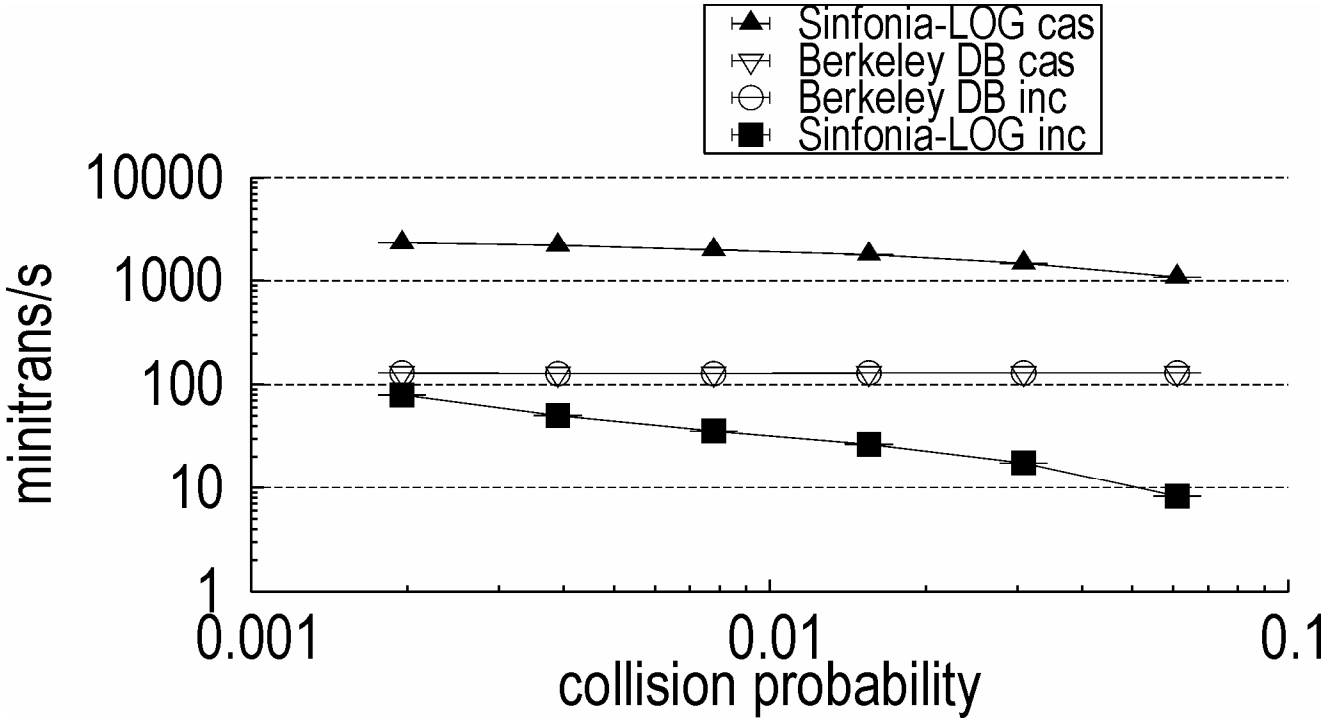
minitransaction spread=all memory nodes (increases with system size):

no scalability

sinfonia service: contention

two workloads

- compare-and-swap (cas) – direct minitransaction support
- increment (inc) – requires caching and retrying (optimistic concurrency control)



hp invent

sinfonia: ease of use

software engineering metrics

	sinfoniaFS	linuxNFS	sinfoniaGCS	spread toolkit
lines of code (language)	3,855 (C++)	5,900 (C)	2,492 (C++)	22,148 (C)
develop time	1 month	unknown	2 months	years
major versions	1	2	1	4

sinfonia: ease of use

- advantages
  - transactions: relief from concurrency, failure issues
  - no distributed protocols, no timeout worries
  - correctness verified by checking minitransactions
  - minitransaction log useful for debugging
- drawbacks
  - address space is low-level abstraction
  - had to lay out data structures manually
  - had to find efficient layout to avoid contention (data structure design problem)

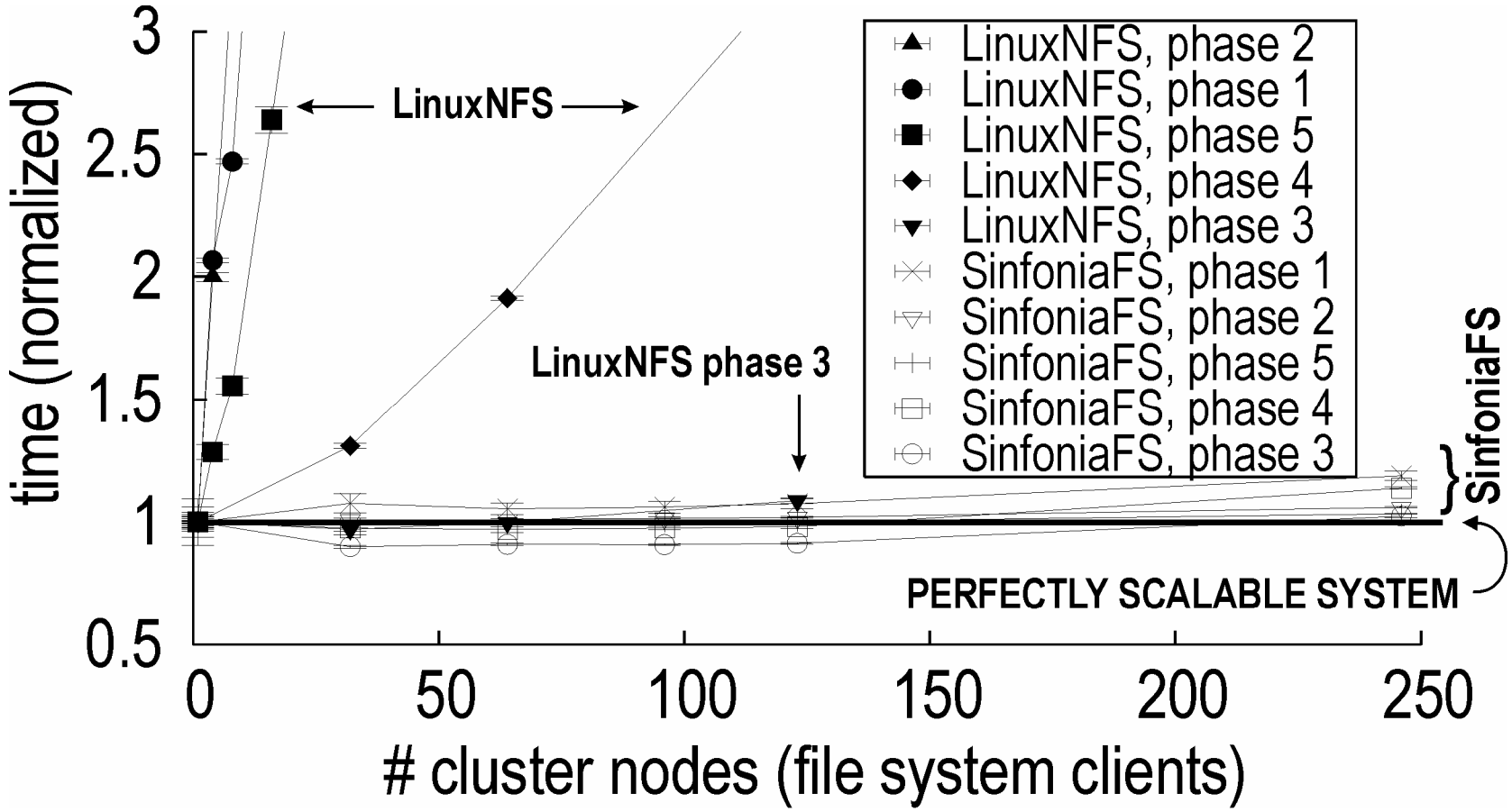
hp invent

sinfoniaFS: base performance

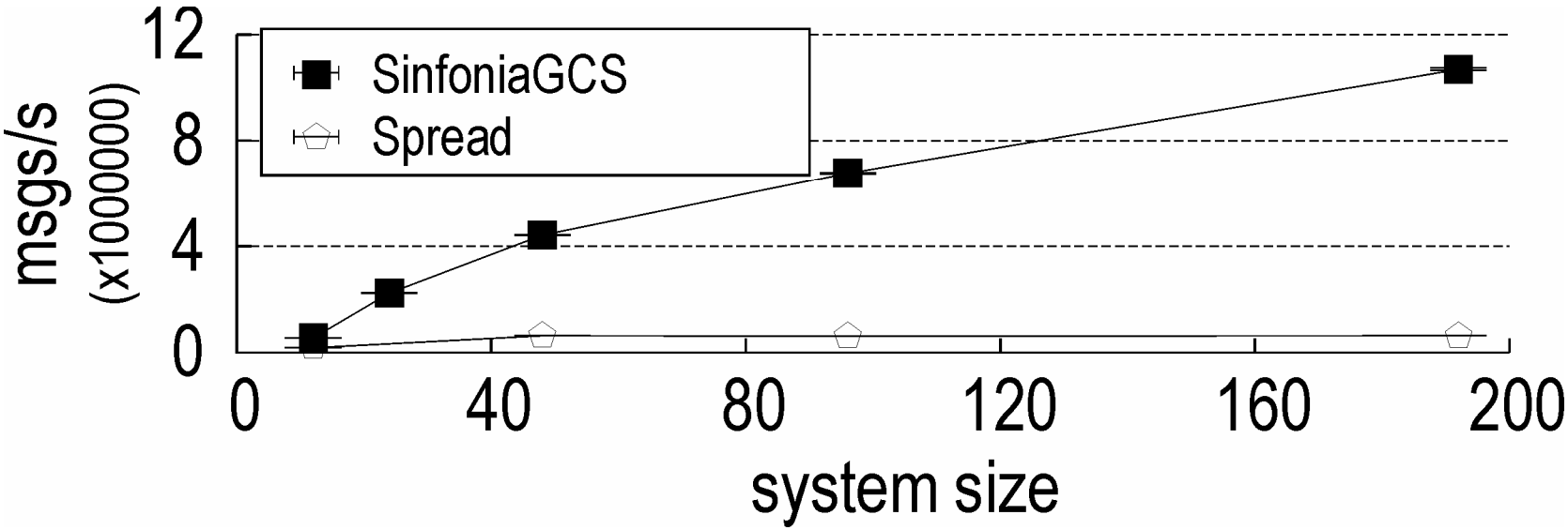
- first considered 1-memory-node system
- benchmarks
  - modified andrew (tcl source code)
  - connectathon nfs testsuite
- sinfoniaFS performs as well as linuxNFS (details in paper)

hp invent

sinfoniaFS: scalability



sinfoniaGCS: scalability



hp invent

related work

- database systems
- distributed shared memory
  - lots, plurix [fakler et al 2005], perdis [ferreira et al 2000]
- camelot, coda
- mime [chao et al 1992]
- berkeleyDB
- thor [liskov et al 1999]
- sdds [gribble et al 2000]
- boxwood [maccormick et al 2004]
- stasis [sears, brewer 2006]
- gfs, bigtable, chubby, mapreduce [200x]

hp invent

conclusion

- sinfonia: a scalable data sharing service for building distributed applications
- main characteristics
  - unstructured address spaces: no unnecessary structure
  - streamlined minitransactions
- general paradigm: built very different apps with it
- main benefits
  - minitransactions hide complexities of concurrency and failures (while providing good performance, fault tolerance and scalability)
  - no protocols to worry about

hp invent