CSC2231: A Case for NOW

http://www.cs.toronto.edu/~stefan/courses/csc2231/05au

Stefan Saroiu

Department of Computer Science

University of Toronto

Administrivia

- **Research report:**
  - If you don't have a group, see me after class
  - Choose topic <-- DUE on Monday at noon!
- **Project:**
  - Form group <-- DUE on Monday at noon!

- **If you are debating whether to take the class**
  - My advice: Don't take it!!!

Playfield

- **Supercomputers (Cray)**
  - Engineered and tuned for performance
- **Massively parallel processors (CM-5)**
  - Commodity processors, custom interconnect, integration, OS
  - Typically NUMA (each CPU has own memory, OS)
- **Symmetric multiprocessors (SUN Enterprise, SGI machines)**
  - Commodity processors, custom integration
  - Shared memory, commodity SMP-aware OS
- **Clusters/NOW**
  - Commodity nodes, OS, LAN
  - Custom applications, glue, network components

Cray

FOR SALE: Cray Y-MPC90

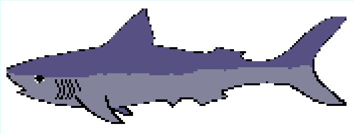
- **In 2000, on eBay:**
  - There is a Cray Y-MP C90 supercomputer for sale on eBay. The current bid as this is written is US$44,500.69. The system features 16 processors, 4 GB of main memory, 4 GB of solid state storage, and 130 GB of RAIDed hard drive space. The original price in 1991 was $10 million.

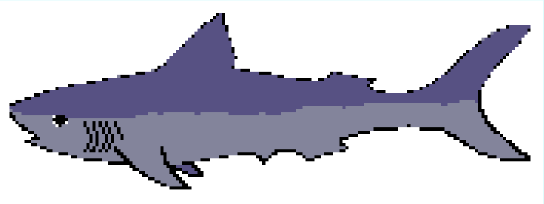
CM-5

SUN Enterprise 450

Cluster

Hardware food chain





workstation/PC:

O($2000)

O(millions) sold

lowest perf./unit

best price/perf.

MPP:

O($1 million)

O(1000s) sold

medium perf./unit

medium price/perf.

supercomputer:

O($10 million)

O(100s) sold

highest perf./unit

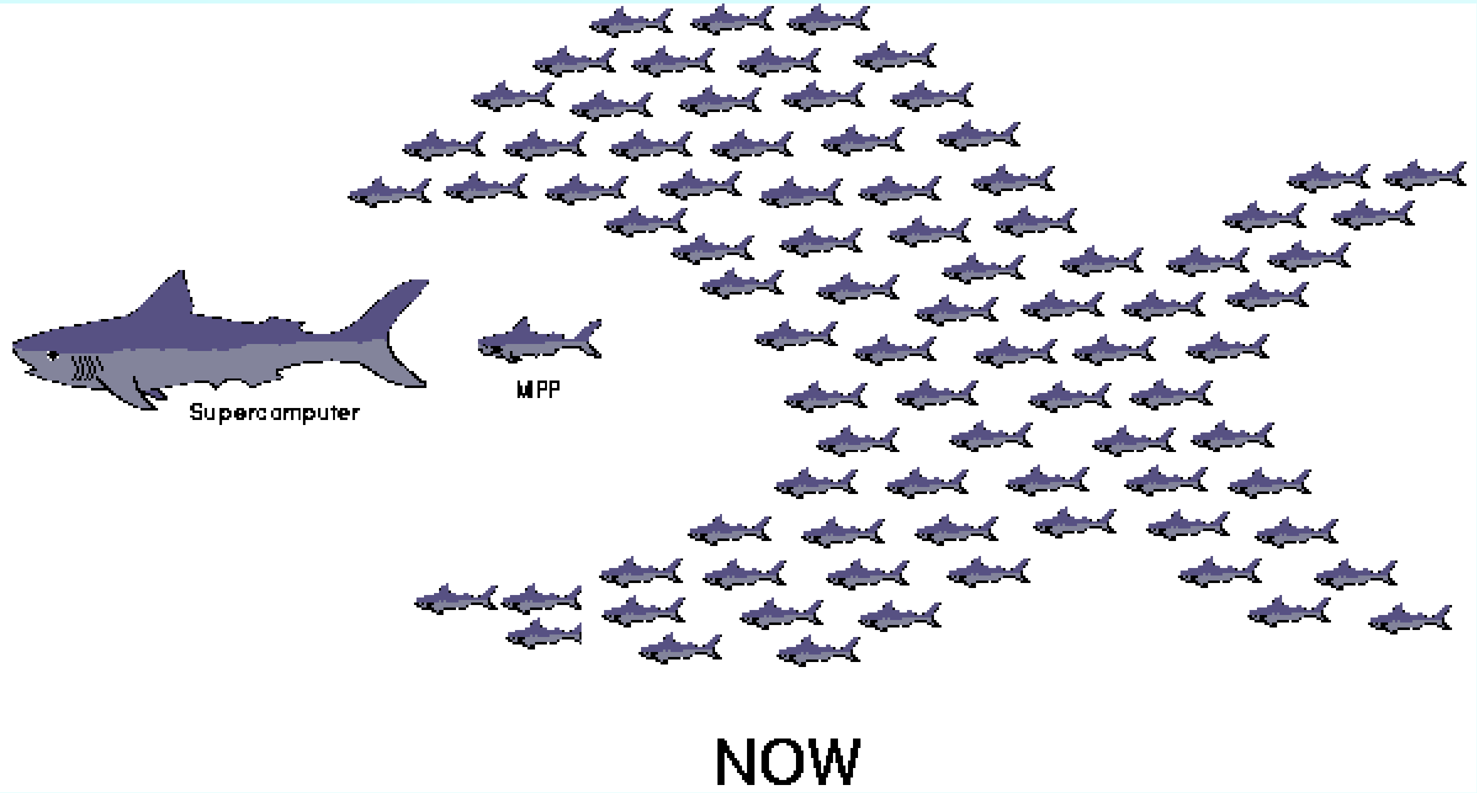
worst price/perf.

Explaining price/performance

	hardware	integration	OS	market lag	applications
supercomputer	custom	custom	custom	3-4 years	customized
MPP	commodity	custom	modified	1-2 years	customized
workstation	commodity	commodity	commodity	none	commodity

- **SCs / MPPs lag workstations in the technology curve**
  - more expensive custom hardware
  - time consuming integration
  - costly, time consuming software development
- **Can one get performance of SC / MPP, with commodity PCs?**

The Now Idea

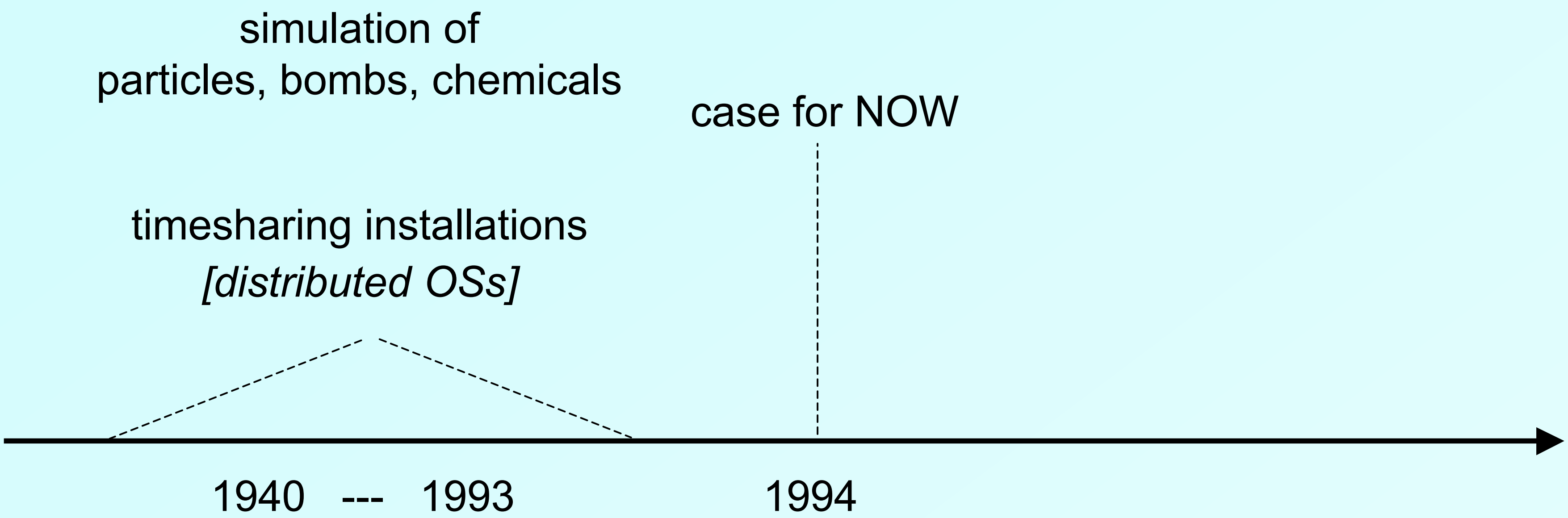


- **hey, let's build a SC / MPP using of commodity PCs**
  - best price / perf., avoids market lag of components

What makes this hard?

- **high bandwidth, low latency, scalable network fabrics**
  - crucial for fine-grained parallel apps, latency sensitive apps
  - fortunately, commodity LANs began to catch up ~1995
    - Myrinet → 100 Mb/s switched ethernet → Gigabit ethernet
- **need to scale and increase performance in OS**
  - NOW cannot afford to fork off of commodity OS development
  - insight: build “glue layer” for unix (GLUnix) (also what Google does)
- **even so, some differences are still visible**
  - architectural: shared nothing, partial failure, heterogeneity
  - software: commodity network stack, multiple OSs

Timeline of SC/parallel apps



- **"setting a research agenda" 101:**
  - predict new apps for emerging technologies
    - rarely get this right
  - or, project old apps onto new technology trends

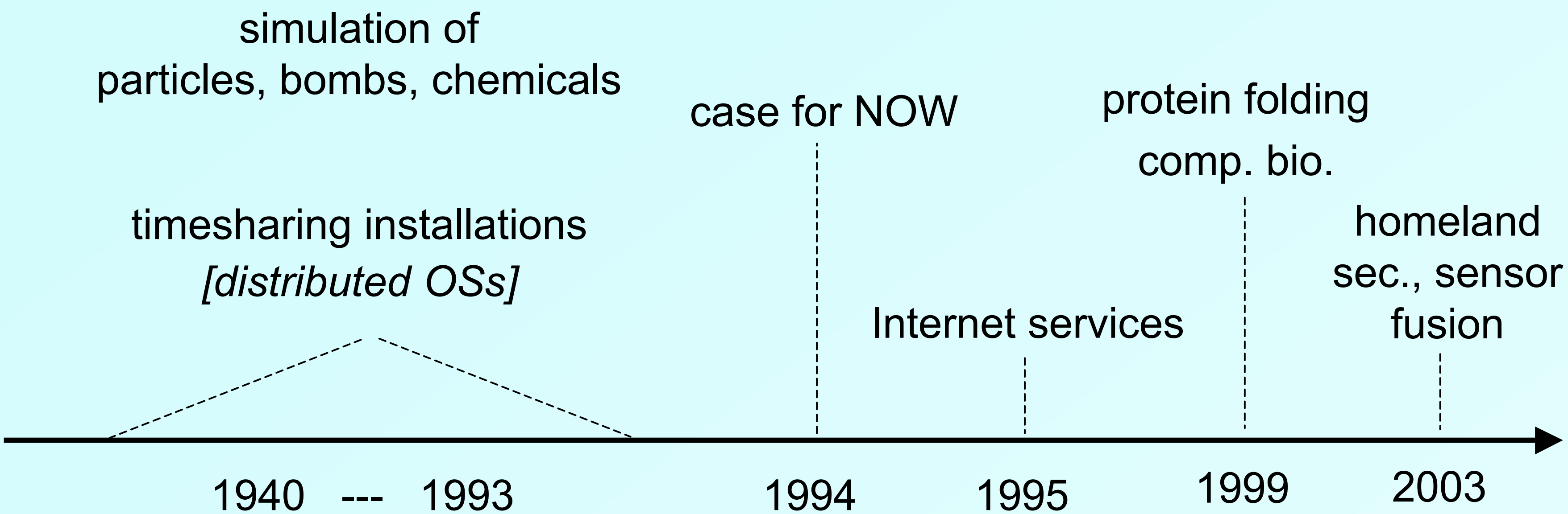
Projecting apps onto NOW

- **one installation to handle interactive and parallel jobs**
  - interactive: lots of idle resources, absorb for parallel jobs
    - harvest desktop PCs into NOW
  - parallel: hogs, need to displace when interactive returns
- **PC is unit of scaling: all resources scale up with cluster**
  - exploit extra resources, idle resources
    - network RAM
    - cooperative caching
- **network latency crucial for fine grained parallel apps**
  - get the OS network stack out of the way
  - user-level networks (begat VIA, infiniband, DAFS, …)

Were these good bets?

- **mostly, but some surprises along the way…**
- **shared nothing is a mixed blessing**
  - + incremental scalability
  - + fault tolerance compared with SCS, MPPs
  - – but, partial failure is a nasty mess
    - the R in RAID: # failures scales with # nodes
      - group membership, replicated data, … [vaxclusters, parallel DB]
      - why programming parallel software is incredibly hard
- **system administration costs can scale up too**
  - if not careful, cost proportional to # nodes
- **PC lifecycle creates significant heterogeneity**
  - HW balancing, network & CPU load balancing, capital depreciation

Rest of timeline of parallel applications



- **A year after the paper was written, everything changed**
  - of course, this was basically impossible to predict
  - but it turned out to be a perfect fit for clusters

Implications of this (next week)

- **dedicated clusters**
  - forget about interactive jobs, distributed OSs
  - cluster as virtual server: three-tier model
    - L4 switch, web server FE, middleware, DB back-end
- **easier programming models**
  - “embarassingly parallel”
  - “Internet consistency”, “reload consistency”
- **different physical packaging**
  - machine room: high density, low futz
  - real estate and energy became real costs (in silicon valley)
- **manageability more important than performance**
  - people are the most expensive resource

Paper's contributions

- **non-contributions**
  - inventing clusters, distributed operating systems
  - inventing (or predicting!) scalable Internet services
    - follow-on work [Brewer et al.] did go after this
- **quasi-contributions**
  - practical user-level networking for clusters
  - notions of cooperative caching in VM, FS, …
- **real contributions**
  - recognizing that commodity LANs are cluster enablers
  - putting another nail in the coffin for MPPs
  - predicting enormous rise in popularity of clusters
    - got simulation app right, initially missed Internet apps

Discussion…

Discussion

- **Is xFS the right model?**

Discussion

- **is xFS the right model?**
  - some traction, but not all the reasons are in the paper
- **similarity between xFS and P2P**
  - both shun any centralized dependency
  - P2P: legal   xFS: practical issue for building-wide FS
- **don't always believe in centralized scaling bottlenecks**
  - don't need to be fully serverless to get benefits
    - parallelize dominant cost to scale performance
      - disk I/Os, disk capacity
    - centralize control structure, make pairwise redundant
- **today's version of xFS: cluster file systems, SANs**

Discussion

- **Will network RAM work in practice?**

Discussion

- **Will network RAM work in practice?**
  - yes, with strong caveats
  - still much worse latency than real RAM
    - 10 ns vs. 50 microseconds
  - partial failures nail you
    - back to MPP failure mode
  - requires free memory on some nodes
    - multiplexing argument: what happens as utilization grows?

Discussion

- **“security to be achilles’ heel of NOWs” – true?**

Discussion

- **"security to be achilles' heel of NOWs" – true?**
  - not true anymore for closed machine room, single app cluster
    - but, clusters-on-demand, cluster reserves, emulab
    - safely time and space division multiplex cluster
  - clusters are a nice multiplicative constant for attacks
    - break into one, you can break into all – zombie army

Discussion

- **what about blades?**

Discussion

- **what about blades?**
  - unit of field replacement is smaller than workstation
    - + hot-swap failed component rather than entire node
    - + cabinetry is better designed (no visible wires, headless)
    - – pay packaging cost per disk, CPU, NIC rather than per node
    - – currently, components lag PCs by about a year
  - higher density
    - + less real estate cost
    - – power density and cooling requirements beyond data centers

Discussion

- **limits of scale of NOWs/clusters?**

Discussion

- **limits of scale of NOWs/clusters?**
  - Google: 20,000 PCs (old figure)
  - Bomb supercomputers: 10,000 PCs
- **all of the benefits at O(100)**

Discussion

- **Did cooperative caching have legs?**

Discussion

- **Did cooperative caching have legs?**
  - useful if:
    - cluster is multiplexed, we're in a load valley (idle resources), and one program can scale up to absorb the idle
    - multiple consumers that share data
      - implicit or explicit sharing
      - by partitioning shared piece, each node only manages its own non-shared, plus a small fraction of shared
      - good idea if non-shared working set smaller than cache size
  - this idea came up later in the Internet services world
    - LARD: partition working set instead of replicating it