P2P, DSM, and Other Products from the Complexity Factory

Willy Zwaenepoel

EPFL
Impact of Research

Not so great

- Many research ideas have lost out
- Many non-research developments won out
Impact of Research

Not so great

- Many research ideas have lost out
- Many non-research developments won out

Why is that?

- We make things too complex
- Note: not: things are too complex
Impact of Research

- Not so great
  - Many research ideas have lost out
  - Many non-research developments won out

Why is that?

- We make things too complex
- Not: things are too complex

Why?

- Publishing/reviewing pushes us to complexity
Apologies, Caveats and Excuses

- Talk is rather polemic in nature
  - ... things are said a little crassly
- Now a dean – intellectual life prohibited
  - “There was once a dean who was so dumb, that other deans actually started noticing it”
P2P

- Peer-to-peer
- No (central) server
- Easier to operate, maintain, scale, make more reliable ...
- Started as an application
- Proposed as an infrastructure for a large number of applications
Research on P2P

- Concentrated largely on DHTs
- Log(n) access
- Chord, Pastry, ...
- Applications: backup, streaming, ...
The Problem with P2P

- Very little application other than illegal file sharing
Reality Check

If we have learned anything about distributed computing over the last 25 years, it is that anything distributed is harder than anything centralized.
Reasons for Distribution

You cannot handle it in one place
  - Performance – controlled replication
  - Availability – controlled replication

Geographical distribution
  - Google!

Illegality – P2P
  - From Napster to Gnutella, Kazaa, …
  - “Raw” traffic numbers are high
  - Much of it static
  - Could be handled by conventional replication (?)
Difficulties for P2P

- Hard to find anything
- Hard to make anything secure
  - Open invitation to attack
  - Actively used by RIAA (pollution attacks)
- Hard to write anything
Advantages for P2P Research

- Complex to find anything
- Complex to make anything secure
- Complex to write anything
Advantages for P2P Research

- Complex to find anything
- Complex to make anything secure
- Complex to write anything
- Complexity begets papers
- P2P = Paper-to-Paper
There are Applications

- Large file multicast
- Can be handled by very simple techniques
  - BitTorrent
- It should worry us that these come from non-research corners of the world!
DSM

- Distributed shared memory
- Parallel computing on clusters
- Distributed memories abstracted as a single shared memory
- Easier to write programs
- Usually by page faulting
- TreadMarks (ParallelTools)
Reality Check

- Clusters are only suitable for coarse-grained parallel computation
- A fortiori true for DSM
Problems with Fine-Grained DSM

- Expensive synchronization
- Expensive fine-grained data sharing
  - Smaller than a page
    - False sharing (can be solved)
    - True sharing
Advantages for DSM Research

- Complex fine-grain synchronization
- Complex fine-grain data sharing
  - Compiler, language, runtime, …

- Complexity begets papers …
(Almost) every paper or grant for research on fine-grain DSM was accepted.

(Almost) every paper or grant for research on coarse-grained DSM was rejected.

It turns out that for real applications a page is not large enough!
Coarse-grain Applications

- Large (independent) units of computation
- Large chunks of data
  - 1 page = 4k
  - Not very large at all
  - Page faulting brings in one page at a time
  - Message passing brings in whole data segment at a time (> page)

- Can be and was done with DSM
  - Increase page size (!!)
  - Compiler support
Competition is Message Passing

- MPI (Message Passing Interface)
- Low abstraction
- No room for complexity fabrication
- As a result more successful
- It should worry us that MPI did not come from distributed systems research but from linear algebra!
Server Performance

At the beginning of the Internet boom, server performance was badly lagging.

Multithreaded or multiprocess servers

- Context switching
- Locking

Two types of solutions

- Exokernel
- Event-driven servers
Event-Driven Servers

Events
- Incoming request, i/o completion, ...

Single thread, event loop

Event handler per event
- Straight code (no blocking)
- At end:
  - nonblocking or asynchronous i/o
  - create (hand-made) continuation
Advantages

- No multithreading
  - No context switching
  - No locking (at least on uniprocessor)
- Control over order of event handling
  - Not bound by OS scheduler
Flash

- Most popular event-driven Web server
- Combined multithreaded / event-driven
- Many follow-ons
- iMimic Networking
It’s too complex
Maybe Ph.D.s can figure it out
Your average industry programmer cannot
Actually, most Ph.D.s can’t either
Many (expensive) bugs
How the Problem was Solved

- Linux O(1) thread scheduler
- Linux futex
  - User-level locking
  - No overhead if no contention
- Benefits of event-driven remain
- But too small to warrant complexity
How the Problem was Solved

- The main servers are all process-based or thread-based (Apache, MySQL)

- It should worry us that these servers did not come out of research!
Painful Observations (1)

- Most of the strong research trends have not found much application
- Non-research designs have won out
- Has to do with this fabricated complexity
Painful Observations (2)

Has to do with publishing/reviewing

- Simple papers tend to get rejected
- Complex papers tend to get in
Your Average Review Form

- Novelty
- Excitement
- Writing
- Confidence
Some Questions to Add?

- Does the added functionality justify the increase in complexity?
- Does the performance improvement justify the increase in complexity?
- Could this system be maintained by an above-average programmer in industry?
- Does this paper simplify a known solution to a worthwhile problem?
Some Likely Review Comments

« Incremental »
« Engineering »
« Nothing new »
« Boring »
It IS Possible

- Virtual machines
- Provide simple solutions to real problems
  - Server consolidation
  - Migration
Virtual Machines

- Virtual machine monitor
- VMM provides a number of VMs
  - IBM VM
  - VMWare
  - Xen
    - Open-source
    - Paravirtualization (VM ~ machine)
Provenance

DISCO: a very complex OS for SMPs

VMWare:
- *Simplified* to Linux/Windows on one machine
- Precise virtualization on x86 very complex

Xen
- Paravirtualization to improve performance and *decrease complexity*
  - VMM less complex
  - Guest OS (slightly) more complex
  - Performance better (?)
The Way of All Technology

All technology
- Becomes more complex on the inside
- Becomes less complex on the outside

Example: car, Windows (?!)

Not sure it fully applies to software
- Most complex systems ever built
- Rare example of discrete complex system
- Maybe we are over the limit already
Nonetheless

Success = interfaces defined early?

Very successful systems
- Apache, MySQL, MPI, VMWare, Xen
- Interfaces stable (few iterations)
- Internal complexity grew

Less successful systems
- DSM, event-driven
- Interfaces unstable, complexified
Standardization (!?)

- I am afraid some of it is necessary
- Find a way through publishing system
Other People’s Advice

Lampson: « Keep it simple »
- True, but somewhat impractical

Einstein: « Everything should be as simple as possible, but no more than that »
- Implement functionality at the right interface
- Keep interfaces stable
Lessons

- Brute force often (not always) works
- Our publishing and reviewing system pushes us in the opposite direction
More Lessons

- It is the interface, stupid
- The implementation can be complex
- The interface has to be simple and stable
Thank you