TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute for System Architecture, Operating Systems Group

Experiences with the Amoeba Distributed Operating System (Andrew Tanenbaum, Robbert van Renesse, Hans van Staveren, Gregory J. Sharp)

presented by Bjoern Doebel

Dresden, 2009-08-11

Amoeba

- Distributed OS
  - microkernel
- “potentially distributed among several countries”
- Goals:
  - Transparent for users
  - Secure
  - Fast

System Architecture

- Workstations
  - Diskless terminals
- Processor Pools
- Specialized dedicated servers
- Network gateways
  - “isolate Amoeba from the pecularities of the protocols that must be used over wide-area networks.”

Objects & Capabilities

- Object-based system
-
- Object addressing via 128 bit capabilities
  - 48 bit server port
  - 24 bit local object ID
  - 8 bit rights (== 8 operations maximum!)
  - 48 bit check field (cryptographic capability)

- Object invocation through RPC

Amoeba RPC

- 3 basic system calls
  - get_request() → open wait
  - put_reply()
  - do_operation → RPC call
- 32 bytes header
  - Capability
  - Opcode
  - 8 bytes payload
- + maximum 30 kB additional payload
- Amoeba Interface Language

Processes and threads

- Client-server architecture
- Run OS services in user space
- Non-preemtable threads
  - Run until block
  - Proved hard to program, when programmers have no notion of *when* blocking occurs.

Amoeba's File System

- Amoeba serves as experimentation platform
- Experiment: Build a fast file system

- *"The decrease in cost of disk and RAM memories over the past decade has allowed us to use radically different design from that used in UNIX."*

- Bullet file server
  - Immutable files
  - Local modification, global storage
  - Checkpoints
  - Garbage Collection

Directory service

- Manages mapping between ASCII names and file/directory capabilities.
- Can implement user-based access control similar to UNIX.
- Support for replicated objects
- Can (in theory?) be replaced at runtime.

Amoeba and the network

- Amoeba nodes grouped in *domains*.
- RPC mostly local to the domain.
- Dedicated service to
  - Connect to other domains
  - Export services
  - *"Another important control is the ability to prevent certain processes (e.g., those owned by students) from accessing wide-area services."*
- Local *"client agent"*

Experiences

- UNIX emulation layer
- Parallel Make
  - No Makefiles!
- Parallel Applications
  - Traveling salesman
    - 25% worse than optimal speedup due to communication overhead!
  - Alpha-Beta search

Performance

- Better than everyone else
  - Lower latency than UNIX and Sun RPC
  - Higher throughput than UNIX and SUN RPC
- Also compare themselves to other existing systems
  - Problem: architectural heterogeneity
  - Some others are equally good or a bit better, but run on faster processor
- Mach really sucks.

Lessons learned

- RPC is point-to-point, which is not always intended. → Multicast
- ***"Only 3 system calls!"***
- Still have no idea how large scale system will work together with capabilities.
- Non preemtible threads were a bad idea.
- Not sure, whether virtual memory is needed.

Discussion

- Insecure capability system

- 1990
  - Computing crosses state borders
  - Special-purpose computers
- Today
  - Computing across state borders is a commodity
  - Computing now resolves borders between computers (think: cloud computing, peer2peer networks)

Discussion (2)

- Points at lots of interesting research directions.
  - Securing capabilities
  - Automatic stub code generation
  - Flexible experimentation system
  - Robustness / Reliability / Checkpointing
  - OS support for high performance databases
  - publish/subscribe, reliable broadcast
  - Build system alternatives to Make
  - Parallel computing, message passing, parallel programming languages
  - Synchronous vs. asynchronous communication paradigms
  - Process migration (do we need it?)

"It was our intention to develop a new operating system from scratch, using the best ideas currently available, without regard for backward compatibility with systems designed 20 years ago."