CS294-5 Multiprocessor Networks

Term Project Presentation

Design Issues for Network RAM

Satoshi Asami

November 30, 1994

Organization

- Motivation
- Previous Work
- Policy/Implementation Questions

Why Network RAM?

- Reduce overhead of paging

- Disk is 100,000 times slower than DRAM, network is *only* 200 times slower

- On average, only 20% of the workstations are busy

Why Network RAM? (cont'd)

- Latency
  - $0.05 \sim 0.1\mu$s DRAM
  - $10 \sim 20\mu$s Network RAM
  - $5,000 \sim 10,000\mu$s Disk
- Bandwidth
  - $50 \sim 100$MB/sec DRAM
  - $10 \sim 20$MB/sec Network RAM
  - $1 \sim 5$MB/sec Disk

Previous Work

- Schilit and Duchamp 91
  - Implemented on a network of T5200s
  - Clients make decision on where to put what
  - Central “broker” keeps info on who has free memory, etc.

Previous Work (cont'd)

- Leff, You and Wolf 93
  - Symmetric client/server model
  - Client *broadcasts* request
  - Servers decide *autonomously* what to cache

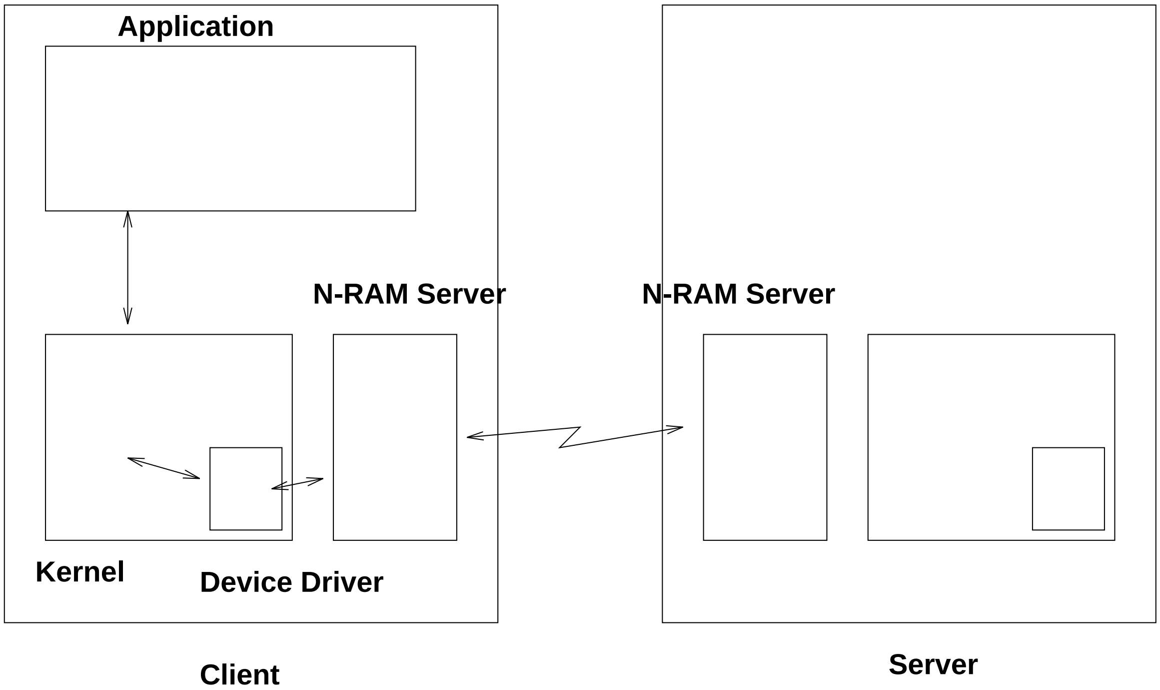
Previous Work (cont'd)

- Comer and Griffoen 90
  - *Dedicated* remote memory servers
  - Disks connected to memory servers
  - Implemented on Sun3s and VAX

Implementation

- Swap device driver calls Network RAM server

- Network RAM server on client calls peer on server



Policy/Implementation Questions

- Client/server model
- Reliability
- Transparency
- Load balancing

Client/server model

- Dedicated server
  - Cost
  - Reliability
  - Scalability

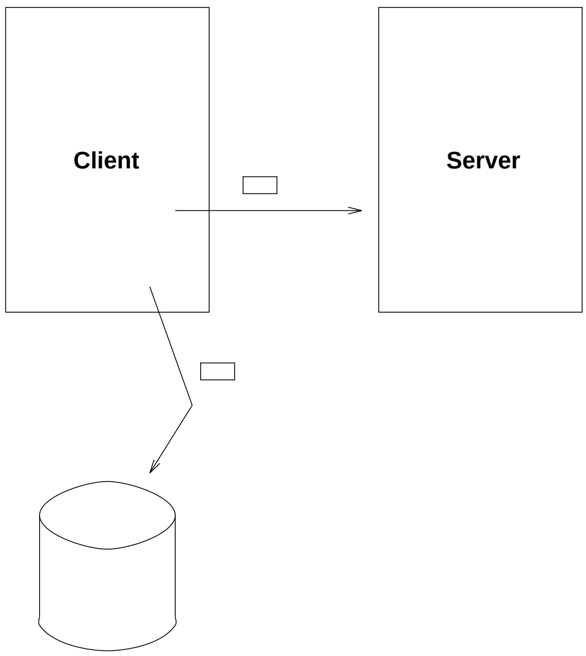
- Symmetric
  - \+ Utilize free memory on network
  - How to find free memory?
    → Use Glunix's idle resource detection service

Reliability

- How to maintain reliability without sacrificing performance?
  - Mirror on local disk
  - Mirror on remote disk
  - Mirror on remote memory
  - Keep parity on remote memory

Reliability (cont'd)

- Mirror on local disk

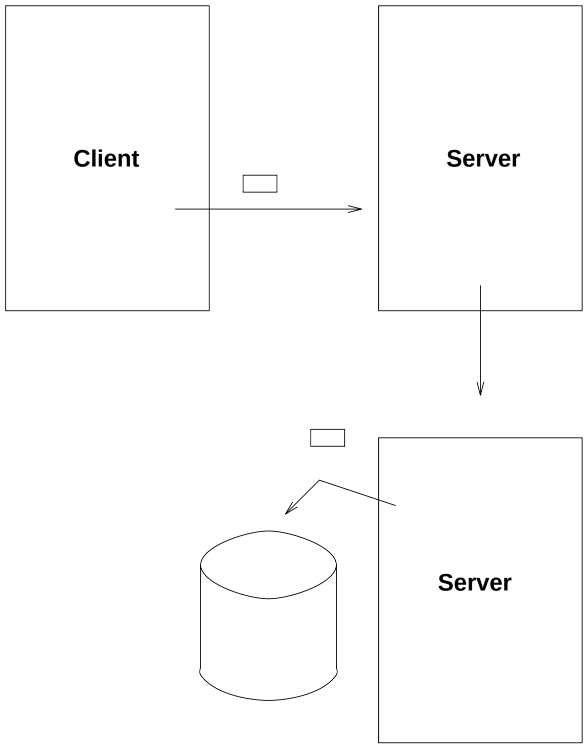


Reliability (cont'd)

- Mirror on local disk
  - \+ Remote machine can throw stuff away without consulting anyone
  - \+ Easy to implement
  - – Speed will be limited to (asynchronous) disk write bandwidth
  - – Another thing for client to do

Reliability (cont'd)

- Mirror on remote disk

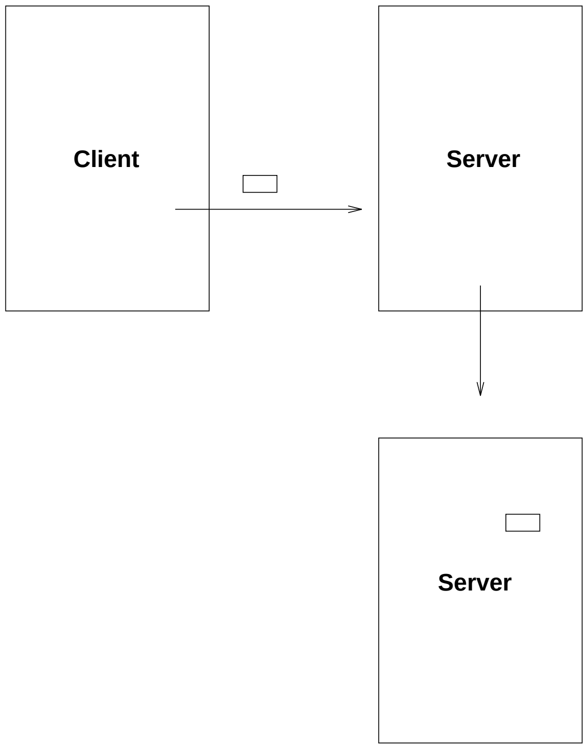


Reliability (cont'd)

- Mirror on remote disk
  - \+ Remote machine can (probably) throw stuff away without consulting anyone
  - \+ Page forwarding can be done by server
  - – Speed will be limited to (asynchronous, parallel) disk write

Reliability (cont'd)

- Mirror on remote memory

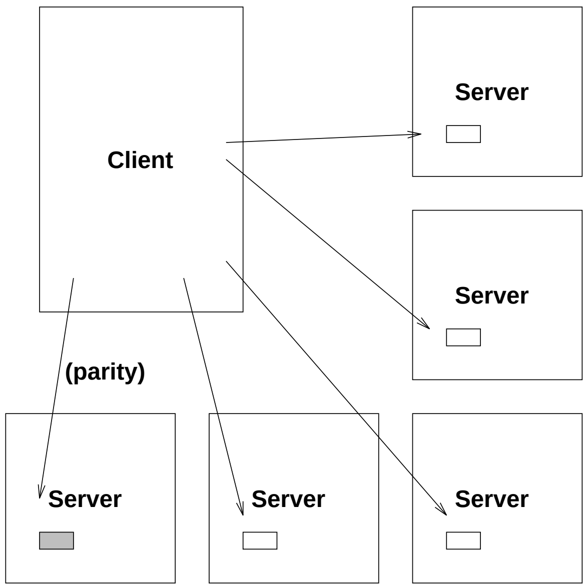


Reliability (cont'd)

- Mirror on remote memory
  - \+ Remote machine can throw stuff away after consulting someone
  - \+ Speed no longer limited by disk
  - \+ Page forwarding can be done by server
  - – Reduces available Network RAM to half

Reliability (cont'd)

- Keep parity on remote memory



Reliability (cont'd)

- Keep parity on remote memory
  - \+ Remote machine can throw stuff away after consulting a bunch of people
  - \+ Requires very little additional memory
  - – Client CPU has to go over all the bytes to calculate parity
  - – Hard to implement

Transparency

- Would the user want to:
  - Recompile the kernel?
    - – Requires kernel source
    - + More control over paging policy
  - Relink the kernel?
    - + Use dynamically loadable modules
  - Recompile the application?
    - – Tedious
    - + More fine-grain control possible

Load Balancing

- How to distribute pages?
  - Random
  - Cyclic
  - Use Glunix's idle resource detection service