

## CS290i - Lecture 19

### Global Computing

Scalable Internet Services and Systems, Spring 2001

Thorsten von Eicken  
Department of Computer Science  
University of California at Santa Barbara

## Idea

- † **Zillions of idle cycles on the internet**
  - † Use idle machines to solve large computational problems
  - † Successful examples:
    - †
    - †
    - †
- † **Build “global computing infrastructure”**
  - † Users with idle machines can join the “grid”
    - † Get paid for cycles
    - † Provide cycles for a “good cause”, e.g. medical research
  - † Organization with large problem can use the grid
  - † Problem: design the infrastructure

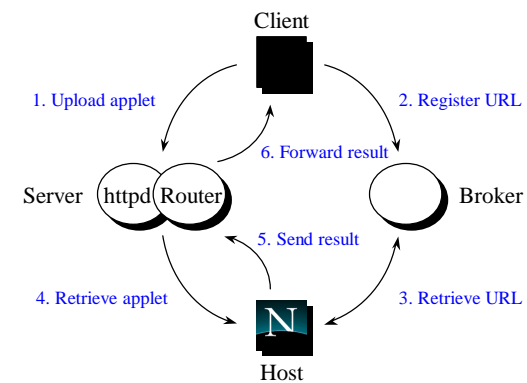
2

## Issues

- † **Cycle provider**
  - † Ease of use
  - † Safety, privacy
  - † Incentive
- † **Cycle user**
  - † Ease of use
  - † Correctness, privacy
  - † Types of applications/algorithms
- † **Broker**
  - † Fault tolerance
  - † Scalability
  - † Security

3

## The Javelin Way



4

## A few projects

- † **SETI@Home**
  - † 3M users (today: 2000), 2 years running (685k CPU years)
  - † Single C app, comm "5 minutes every few days"
- † **Javelin**
  - † Research project @UCSB
  - † Java-based, on-line only
- † **CX**
  - † Research project @UCSB
  - † Java-based, task model
- † **Parabon**
  - † Commercial project (Maryland) , soon for cash
  - † Java-based, off-line ok
- † **Entropia**
  - † Commercial project (UCSD), soon for cash
  - † Win32-based, off-line ok, running FightAIDS@Home, SaferMarkets, ...
- † **United Devices**
  - † Commercial project
  - † Win32-based, outgrowth of SETI@Home

## Getting started

- † **Setting up central servers**
  - † Typical reliable site
  - † Custom brokerage software
  - † Monitoring
- † **Easy-to-install client**
  - † Handling communication
  - † Not disturbing the user
- † **Programming model**
  - † Splitting-up the app, communicating, controller app
  - † Libraries, e.g. to query DBs
- † **Safety**
  - † Technology & social solutions

## Ensuring Correctness

- † **Correctness - Can one trust the result?**
  - † Hosts could return a bogus result
- † **Result verification**
  - † Some results can easily be verified
  - † Statistical methods
  - † Insert checksum computation
  - † Redundant computation by independent hosts
  - † Canary computations
- † **Reputation services**
  - † give hosts an economic disincentive for faking computations

## Ensuring Privacy

- † **Privacy - Can one trust a host?**
  - † Host could spy on your computation
- † **Split computation into N parts**
  - † which individually don't convey any information
  - † need K out of N
- † **Encrypted computing**
  - † Encrypt input and program before sending to host
  - † Host performs computation in encrypted domain
  - † Host sends encrypted solution back

## Encrypted Computing

### † Original Program:

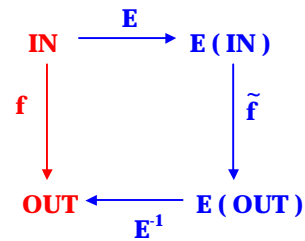
†  $f$ :  $OUT = f(IN)$

### † Encrypted Program:

†  $\tilde{f}$ :  $E(OUT) = \tilde{f}(E(IN))$

### † What can we encrypt ?

- † Input
- † Output
- † Program



## Solving System of Linear Equations

### † Original system of linear equations

†  $Ax = b$

### † Encode the inputs A and b

- †  $TAx = Tb$
- † Solution has not changed

### † Encode inputs and outputs

- †  $TAT^{-1}Tx = Tb$
- † For suitable T stability is not affected (same eigenvalues, conditioning number)

## Status of Encrypted Computing

### † Known:

- † MIN, MAX, Fast Fourier Transform
- † Solving system of linear equations
- † Solving partial differential equations
- † Simple image transformations
- † Simple search problems

### † Not known:

- † Non-linear problems

### † Is there a general theory?

- † See what problems are solvable
- † Characterize encryption overhead

## Safety Issues

### † Code could contain virus

- † Could spy on data
- † Could destroy some data
- † Could destroy machine

### † Denial of service attack

- † Could use too much of some resource
  - † e.g. CPU time, disk space, network

### † Behind a firewall

- † Could attack other machines behind firewall

## Standard Techniques

### † Run applet/application in a logical fault domain

- † memory-safety
  - ‡ can only access allowed memory
- † interface-safety
  - ‡ can only access interfaces provided at time of creation
- † resource-safety
  - ‡ limits amount of resources (CPU, memory, disk)

### † Implementations

- † Operating systems
  - ‡ use hardware protected address space
  - ‡ between applications
  - ‡ between users
- † Java virtual machine (Java applets)

13

## Safety Issues

### † Is Java the solution?

- † Overhead
- † Security policy too restrictive
  - ‡ Requests to database servers
  - ‡ Local file storage

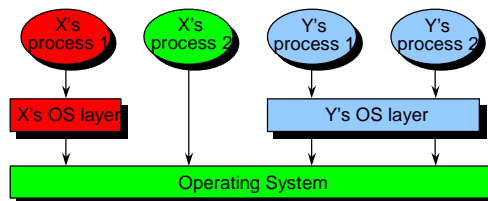
### † Alternatives?

- † Trust-based or inspection-based solutions?
- † Arbitrary binaries in "Confined shell"

14

## "Confined Shell"

- † Intercept all system calls
  - ‡ intercepts selected system calls (similar to strace or truss)
  - ‡ can actually change the behavior of a syscall
  - ‡ can change arguments or results of syscall
  - ‡ can then execute the original system call or implement fake one
- † Can be done at user-level



15

## Safety (cont.)

### † Confined shell

- † Assumes that OS does not have any bugs...
- † Many system calls are multi-purpose
  - ‡ Hard to check
  - ‡ Is there a useful simple subset?

### † What kind of network connections does one allow?

- † do all network connections have to go through a proxy?
  - ‡ e.g. to avoid communication within firewall
  - ‡ e.g. e-mail "death threat to the president"

16

## Broader issues

### † Trusting computation elsewhere

- † Using shared facilities
  - † E.g. "Oracle DB service"
- † Is it really one's own client connecting back?
  - † Updates/upgrades

### † P2P

- † Eliminate central bottleneck
- † DoS/Spam resistance
- † Propagating service quality information

17

## Project 4 presentations

### † 5 minutes per groups max

- † Idea: how did you solve XYZ?

### † Questions

- † Do you use raw servlets, Webmacro, JSP? View/control/logic layering?
- † Do you pool connections to the DB? Anything noteworthy?
- † Do you use DB transactions? DB locks?
- † Do you cache data? Which? How do you invalidate?
- † Do the web servers communicate with each other? Or only the DB? Do you use JavaGroups?
- † Do you batch requests? Circulate tokens? Other concurrency control?
- † How do you trigger settlement of orders? Assign order ids?

18