Design and implementation of a Distributed Virtual Machine for networked computers

Emin Gun Sirer, Robert Grimm
Arthur J. Gregory, Brian N. Bershad
University of Washington

OS/Network Lunch Presentation by
Alexander V. Konstantinou

Introduction

- Future networks characterized by mobile code
- Platform independence
- Wide range of computing devices
  - Desktop, embedded systems, PDAs
- Java, Inferno, …

Current VM Shortcomings

- Manageability
  - No central control
- Performance
  - Processing/memory requirements
- Security
  - Large Trusted Computing Base (TCB)
- Scalability
  - Monolithic architecture does not scale

Paper Goal

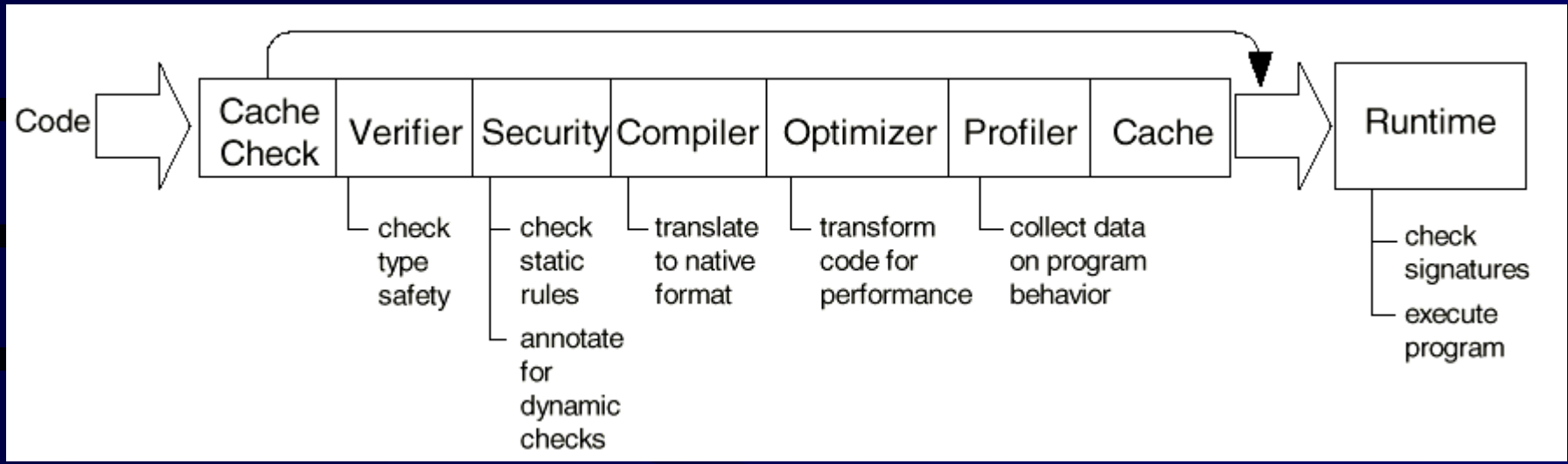
- *Distributed* virtual machine architecture
- Factor VM services into logical components
- Move services from client to network
- Support centralized management of critical VM functions

Paper Overview

- Introduction
- Architecture Overview
- Services
- Performance Evaluation
- Optimizations
- Related Work / Conclusions

DVM Architecture

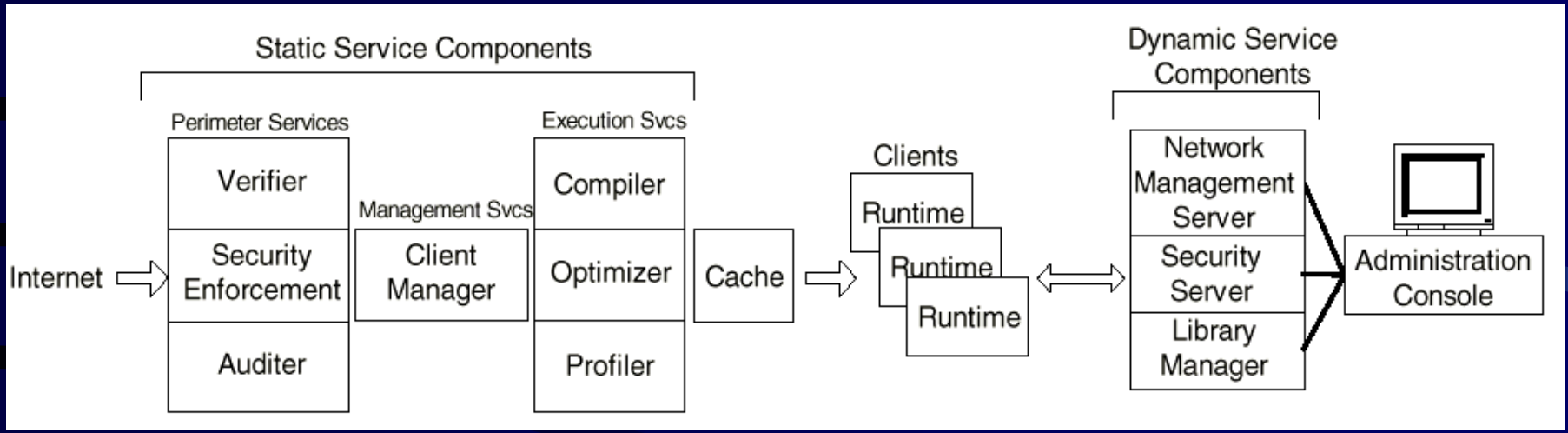
VM Architecture



Distributed VM Architecture

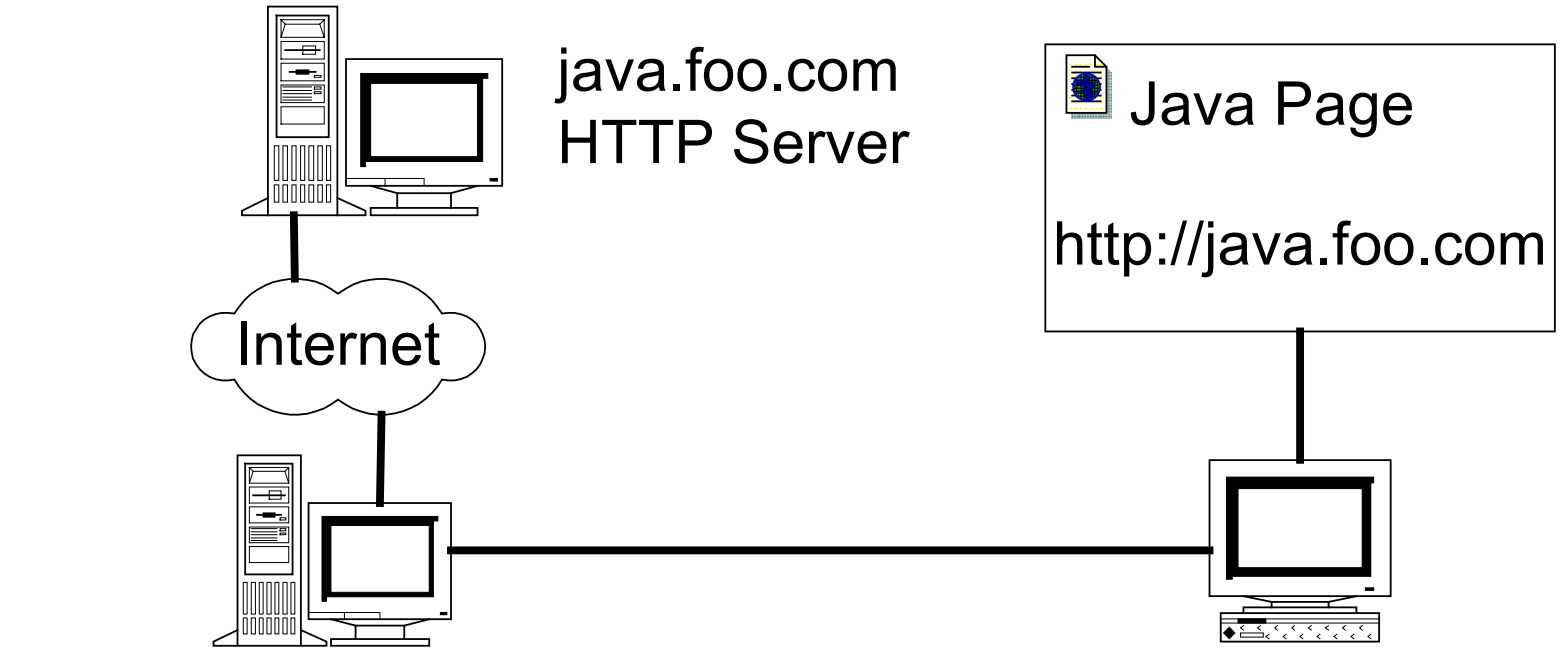
- Distributing VM services :
  - Can't use existing monolithic interfaces
  - Factor functionality into **static** & **dynamic** components
- Static functionality can be moved around
- Dynamic functionality executed at client

Static & Dynamic Functionality



Glue = binary code rewriting

Deployment Architecture



DVM Services

DVM Services: Verification

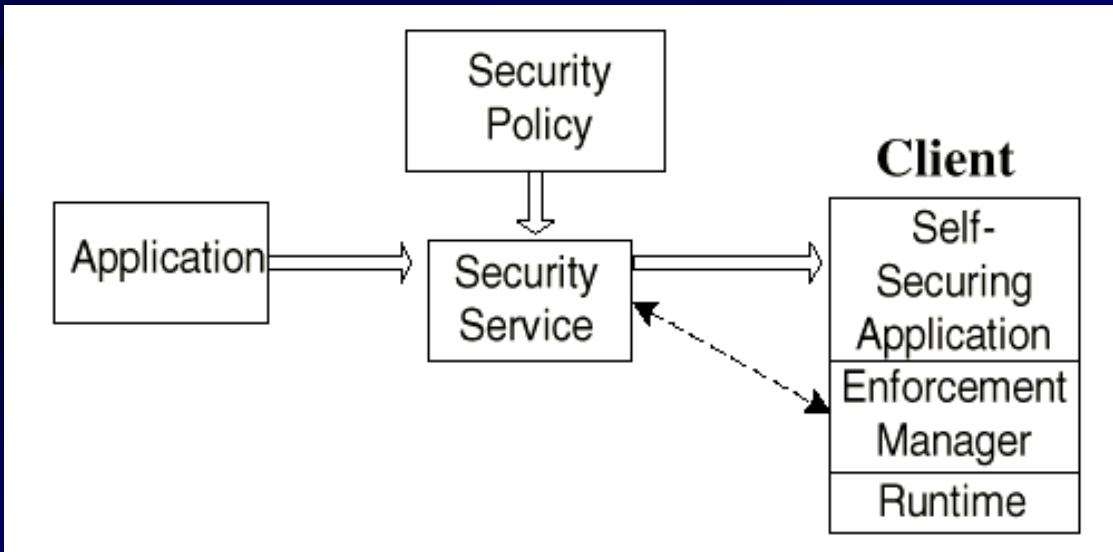
- Monolithic Verifier Issues :
  - Different implementations
  - Security holes cannot be centrally patched
  - Memory/processor requirements (HP dropped it)
- Phases 1, 2, 3 : check class in isolation
  - Statically at server
- Phase 4 : verify imports
  - Dynamically at client (rewriting)

Verification Rewriting

```
Class Hello {
  static boolean __mainChecked = false;
  public static void main() {
    if (__mainChecked == false) {
      RTVerifier.checkField("java.lang.System", "out",
                                    "java.io.OutputStream");
      RTVerifier.checkMethod("java.io.OutputStream",
                                     "println",
                                     "(Ljava/lang/String)V");
      __mainChecked = true;
    }
    System.out.println("hello world");
}
```

DVM Services: Security

- Uniformly enforce organ. security policy
- Central point of policy specification
- Impose checks in **any** part of code



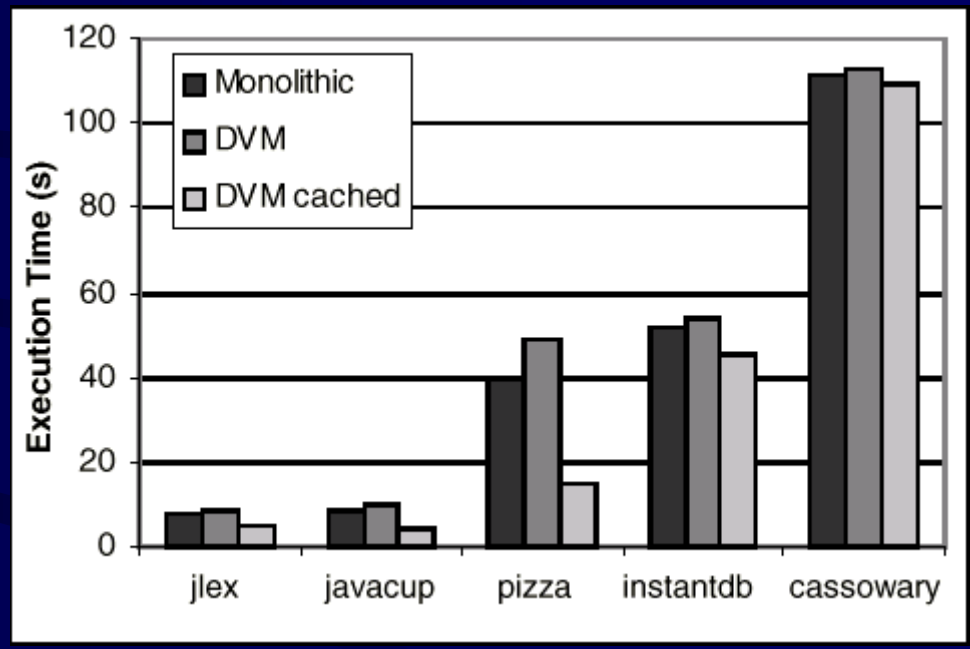
Security Service Highlights

- Centralized policy enforcement
  - DTOS (mach) model
  - XML-based access matrix
- Control of arbitrary resource access
  - Not dependent on app. programmer checks
- During execution query security service
  - cache + invalidation protocol
- Cons: Download an enforcement manager
- *Other services*: remote monitoring/profiling + compilation

Performance Evaluation

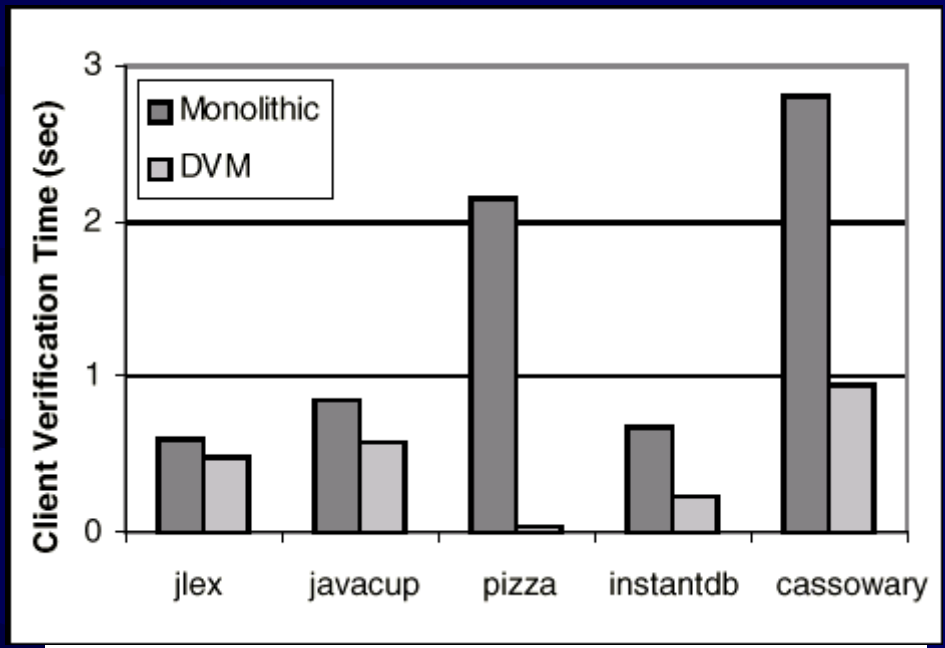
Performance Evaluation

- Sun JDK 1.2
  - Vanilla v.s.
  - Removed monolithic services
- Discard 3/Average 5
- DVM 11% slower
  - Proxy parsing overhead
- Pizza (compiler) ??



Client-Side Overhead

- JDK 1.2 verifier written in C
- Still, DVM clients perform better



Benchmark	Static Checks	Dynamic Checks
Jlex	291679	371
Javacup	415825	806
Pizza	289495	541
Instantdb	1066944	3426
Cassowary	1965538	2346

Security Runtime Overhead

- Check: wall-clock time
- Overhead: checked - baseline
- Download: global security policy overhead

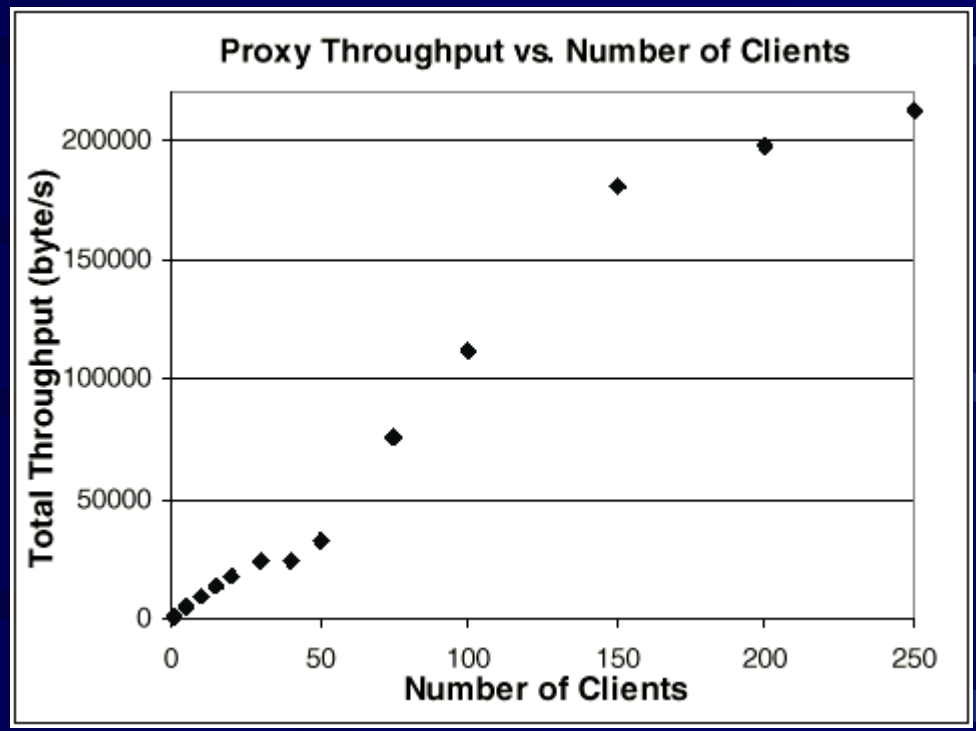
Description	Baseline (no check)	JDK (check)	JDK (overhead)	DVM (download)	DVM (check)	DVM (overhead)
Get Property	0.0020	0.0488	0.0468	5.830	0.0092	0.0072
Open File	1.406	8.631	7.224	6.406	1.430	0.0238
Change Thread Priority	0.0638	0.0645	0.0007	5.026	0.0815	0.0177
Read File	0.0141	N/A	N/A	4.146	0.0368	0.0227

Proxy Overhead

- 100 random applets
- No caching
- Average download 2198ms
  - std dev 3752ms (large)
- Proxy adds 265ms (**12% overhead**)

DVM Proxy Scaling

- Clients 0-250
- 1 -- 1.2 sec/kB latency
- No caching
- Degradation >250
  - Memory (64MB)



Class Download Optimizations

- Class byte-code download :
  - One at a time, or as .jar file
- Not efficient for low bandwidth/high latency
- 10-30% of downloaded code never invoked
- Solution: restructure classes at server based on profiles

Conclusions

- New system architecture (DVM)
- Factor VM services
  - Centralize security services
  - Reduce client requirements
- Performance comparable to monolithic VMs

Issues ?

- Comparison to application servers
- Assume trusted internal network
- Assume single point of entry (proxy)
- Clients using SSL
- Trust issues (re-signing code)
- Scalability (very large corporations)