OmniVM

Efficient and Language-Independent Mobile Programs

Ali-Reza Adl-Tabatabai,
Geoff Langdale,
Steven Lucco and
Robert Wahbe

from Carnegie Mellon University and Colusa Software

presented by Suming Chen
April 16, 1998

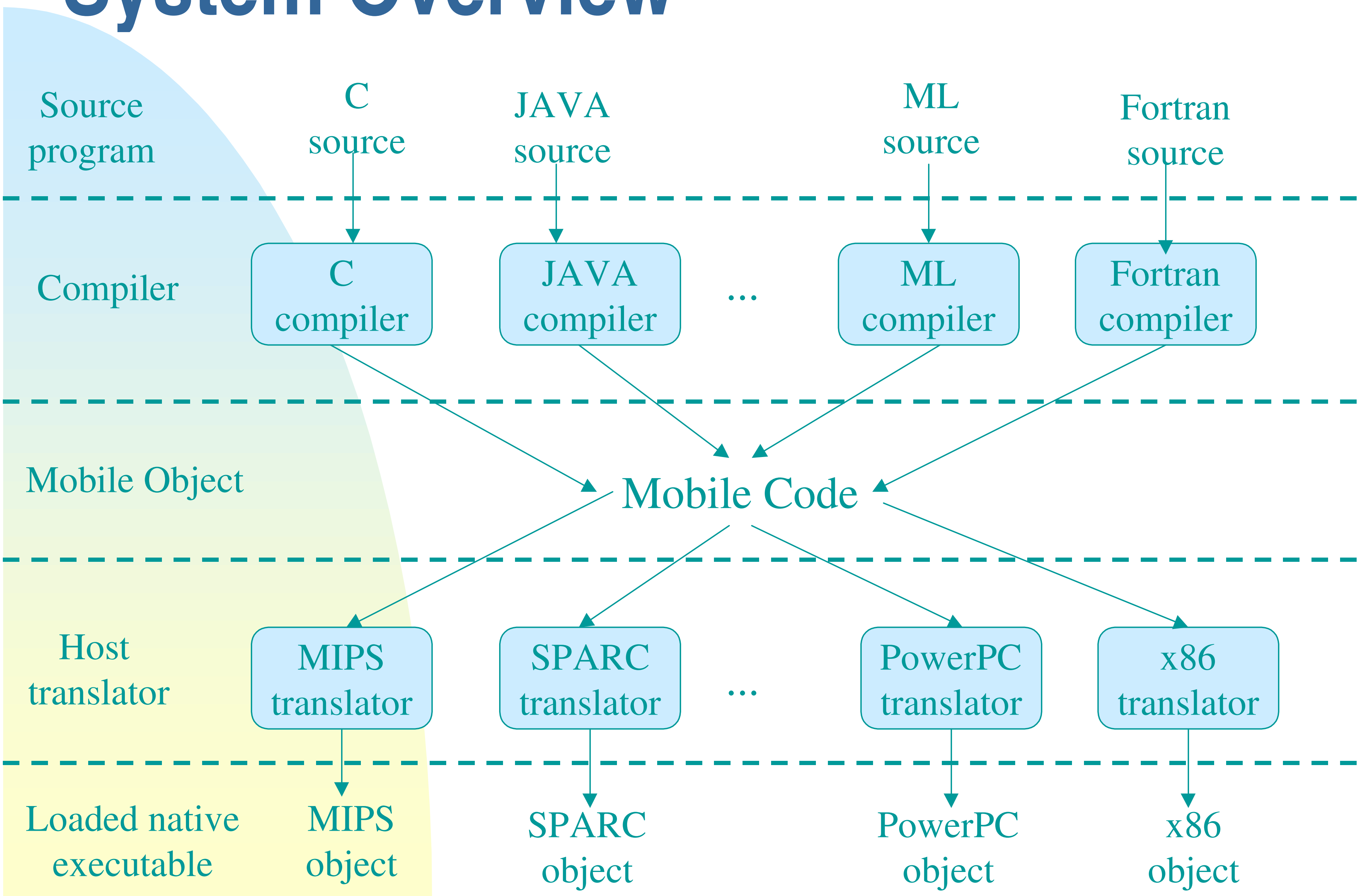
Motivations

- Two ways to ensure safe execution of mobile code:
  - Language semantics
  - Abstract machine interpretation
- Goals
  - Portable
    - run on any machine
    - safe execution of any source language
  - Efficient
    - fast translation
  - Easy to generate code

What is OmniVM?

- A safe, efficient, and language-independent system for executing mobile program modules
- RISC-based design
  - instruction set, register file organization, data format, exception model, and segmented virtual memory model

System Overview



Safety

- Use software fault isolation (SFI) to enforce safety
- Host-imposed permission on address spaces
- Inline checks during load time and translation time
- In case of violation, delivers access violation exception

Performance

- Measured on
  - 90 MHz Intel Pentium based "Precision Pentium" running Windows NT 3.5
  - PowerPC 601 based IBM RS/6000 running AIX version 3
  - Mips R4400 based SGI Indigo-2 machine running Irix 5.2
  - Sparc based SPARC10 running Solaris 2.4
- Use four C programs from SPEC92 suite
  - li, compress, alvinn, and eqntott

Performance *con't* ...

Program	Execution time relative to native							
	Mips		Sparc		PPC		X86	
	SFI	No SFI	SFI	No SFI	SFI	No SFI	SFI	No SFI
Li	1.10	0.91	1.05	1.02	1.18	1.08	1.11	1.10
Compress	1.04	0.96	1.02	1.01	1.23	1.18	1.02	1.02
Alvinn	1.20	1.09	1.07	1.03	1.08	0.97	1.25	1.22
Eqntott	1.20	1.18	1.04	0.99	1.35	1.35	1.06	1.04
Average	1.14	1.03	1.05	1.02	1.21	1.14	1.11	1.10

Execution time of mobile code relative to native code generated by cc

Performance *con't ...*

Program	***Execution time relative to native***							
	Mips		Sparc		PPC		X86	
	SFI	No SFI	SFI	No SFI	SFI	No SFI	SFI	No SFI
Li	1.11	0.92	1.05	1.01	1.04	0.94	1.09	1.09
Compress	0.78	0.72	1.02	1.01	1.08	1.13	1.01	1.01
Alvinn	1.12	1.01	1.08	1.02	1.36	1.21	1.09	1.06
Eqntott	1.04	1.02	1.03	1.01	0.66	0.66	1.05	1.03
Average	1.01	0.92	1.05	1.01	1.03	0.98	1.06	1.05

Execution time of mobile code relative to native code generated by gcc

Performance *con't ...*

Program	*Execution time relative to native*							
	Mips		Sparc		PPC		X86	
	SFI	No SFI	SFI	No SFI	SFI	No SFI	SFI	No SFI
Li	1.18	1.06	1.11	1.07	1.35	1.14	1.18	1.15
Compress	1.04	0.84	1.18	1.16	1.28	1.23	1.09	1.07
Alvinn	1.37	1.20	1.21	1.17	1.32	1.04	1.79	1.71
Eqntott	1.08	1.06	1.24	1.21	1.35	1.35	1.22	1.16
Average	1.17	1.04	1.21	1.15	1.33	1.19	1.32	1.27

Execution time of mobile code without translator optimization, relative to native code generated by cc.

Conclusion

- Including the overhead for enforcing safety, OmniVM executed benchmark programs within 21% as fast as the optimized, unsafe native code
- Fastest system for mobile code
- First to implement safe, mobile code efficiently and language-independently

Final Remarks

- Need to retarget compilers
- Requires machine-dependent translator residing on the host machine and these translators need to have prior information on which segments in the memory are safe for the mobile code
- Current OmniVM does not support read protection