Client Ahead-Of-Time Compilation for Digital TV Software Platform

Sunghyun Hong, Soo-Mook Moon

Virtual Machine & Optimization Lab.
Electrical and Computer Engineering
Seoul National University

Presenter: Hyuk-Woo Park

Client Ahead-of-Time Compilation

- VM is a mainstream for app-downloading systems
  - Java xlets for DTV, Java Android apps, JavaScript-based web apps
- Interpretation is slow, so uses just-in-time compiler
  - However, JIT compiler suffer from compilation overhead
- Proposed client ahead-of-time compiler [Hong et al. LCTES '07]
  - Save JIT-compiled methods on client devices, so we can use them directly w/o compilation overhead
- This paper evaluate it for Digital TV S/W platform

Outline

- DTV S/W platform
- Client Ahead-of-Time Compilation for DTV
- Relocation Issues
- Experimental Results
- Summary

DTV S/W Platform

- DTV allows data-broadcasting based on Java
  - Sending data as well as picture/sound
  - Java xlets from TV station + Java system/middleware on set-top box

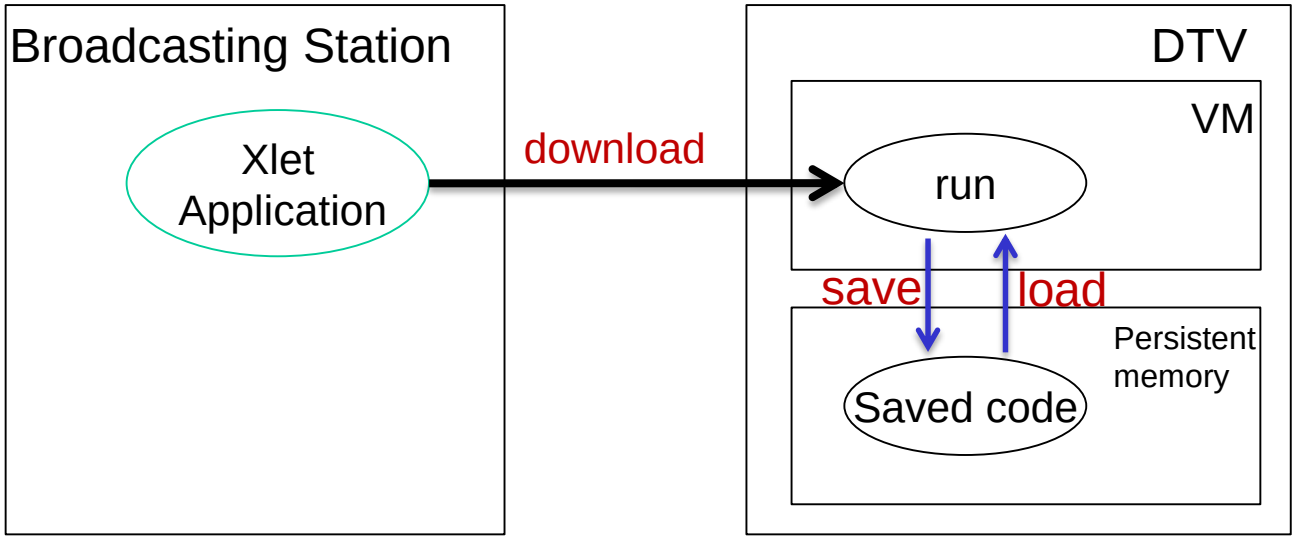
Java Performance and JIT

- Java is slow, so DTV JVM uses JIT compiler
  - Unfortunately, DTV suffers from JIT-compilation overhead seriously, because many methods are compiled but they are not hot
  - Different from benchmarks

➔ We propose using client-AOTC for DTV

client-AOTC for DTV

- Save JITC-methods for reuse in next runs
  - When current run ends, JITC-methods are supposed to be thrown out
  - Instead, we save them for reuse



One Issue: Relocation

- **Relocation**
  - Saved machine code may include address constants that are different for each run
  - c-AOTC must modify (relocate) them before loading
- Depends on JVM and JITC implementation
- In our previous work, relocation was an big issue

Relocation Example of Previous c-AOTC

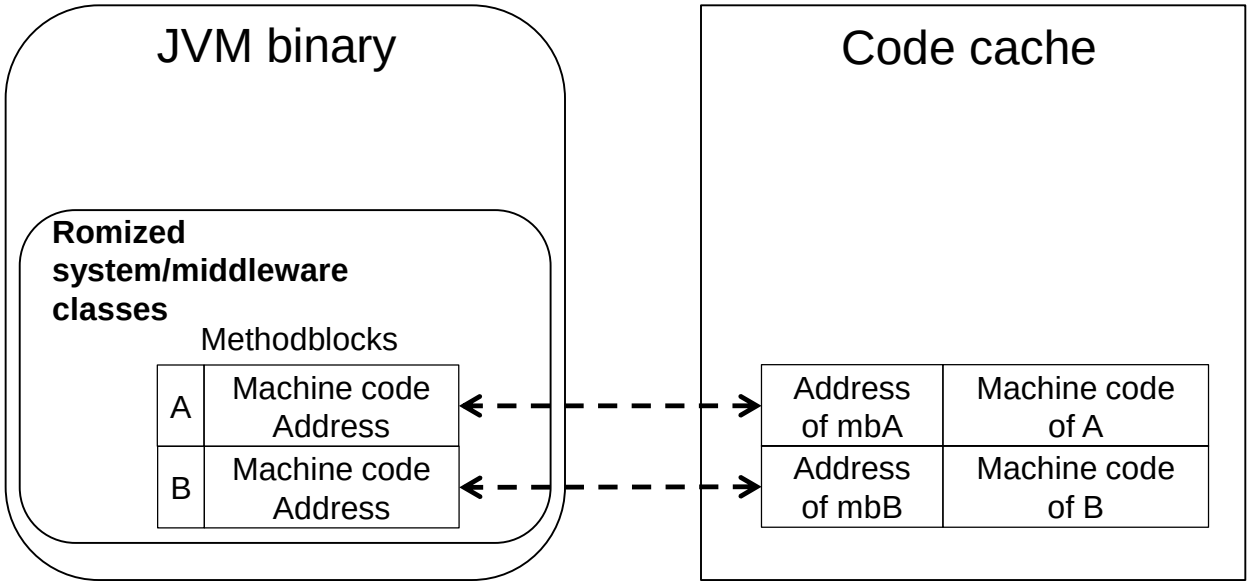
- For example, the machine code for “invokestatic” is

```
#set the address of callee static method
r0 ← half address of a static method
r0 ← another half address of a static method
Jump to r0
```

  - Address of callee method is encoded in machine code
  - When this code is loaded by c-AOTC in the next run, the address is not valid, so we need to relocate it

Relocation Issue for DTV Platform

- System and middleware classes are romized
  - Romized class files composed of class block (CB), method block (MB), and the method bytecode are compiled together with the JVM
  - Previous relocation target is now compiled to machine code indirectly accessing the data structure (e.g., MB), whose address is fixed



Relocation Example of DTV c-AOTC

- For example, the machine code for “invokestatic” is

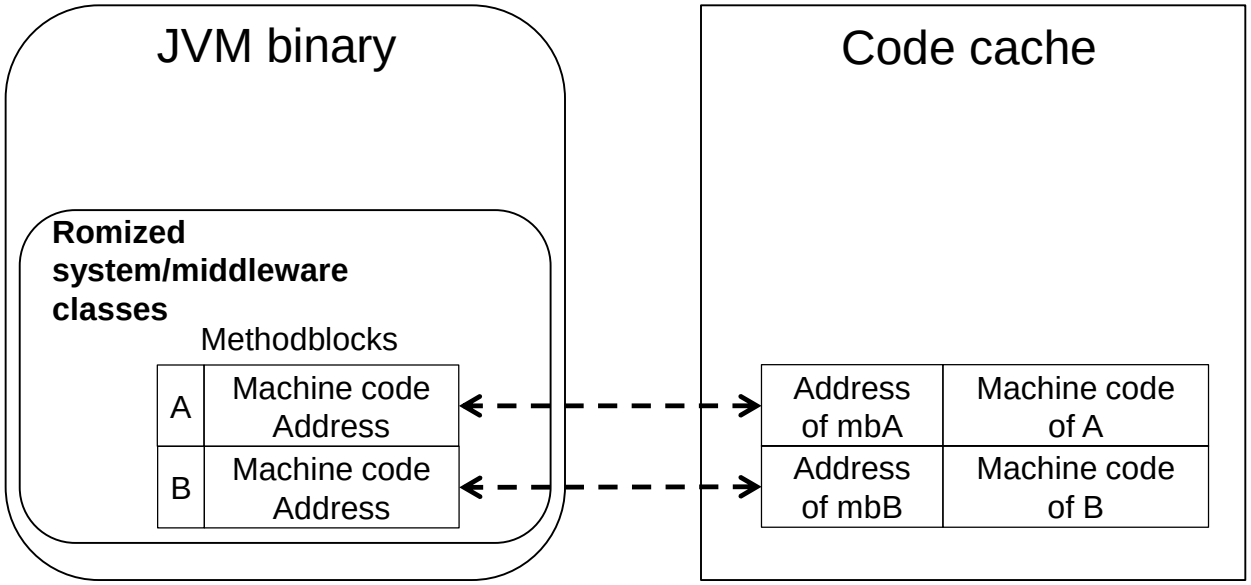
```
r0 ← high address of methodblock of static method A
r0 ← low address of methodblock of static method A
#get the machine code address of method A
LW s7, 0(r0)
Jump to s7
```

  - Address of callee methodblock is encoded in machine code
  - When this code is loaded by c-AOTC, the address is valid

➔ No modification is needed

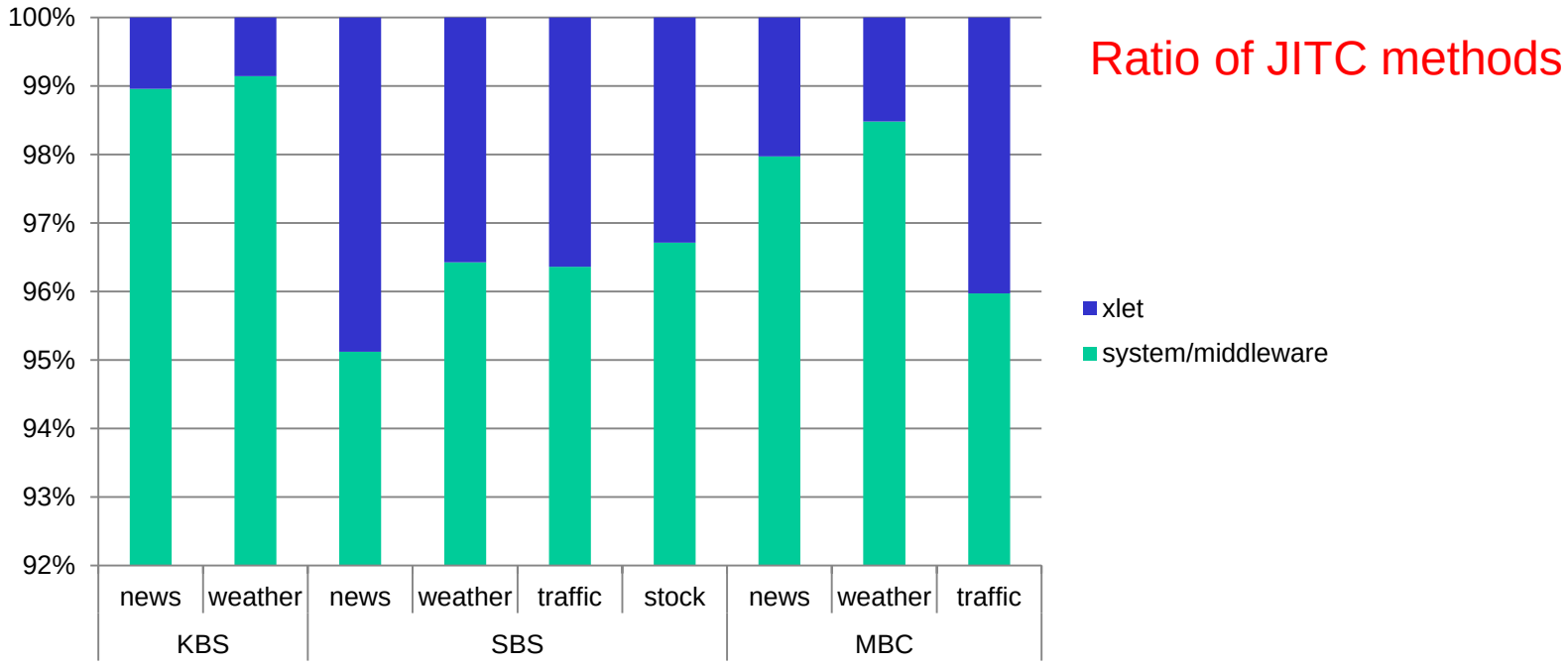
Relocation for System/Middleware

- No modification of encoded address constant
- But, address of method block needs to be updated



Relocation for xlet Methods

- Downloaded xlet methods are not romized, so they require relocation, but they are rarely executed, thus rarely JITCed

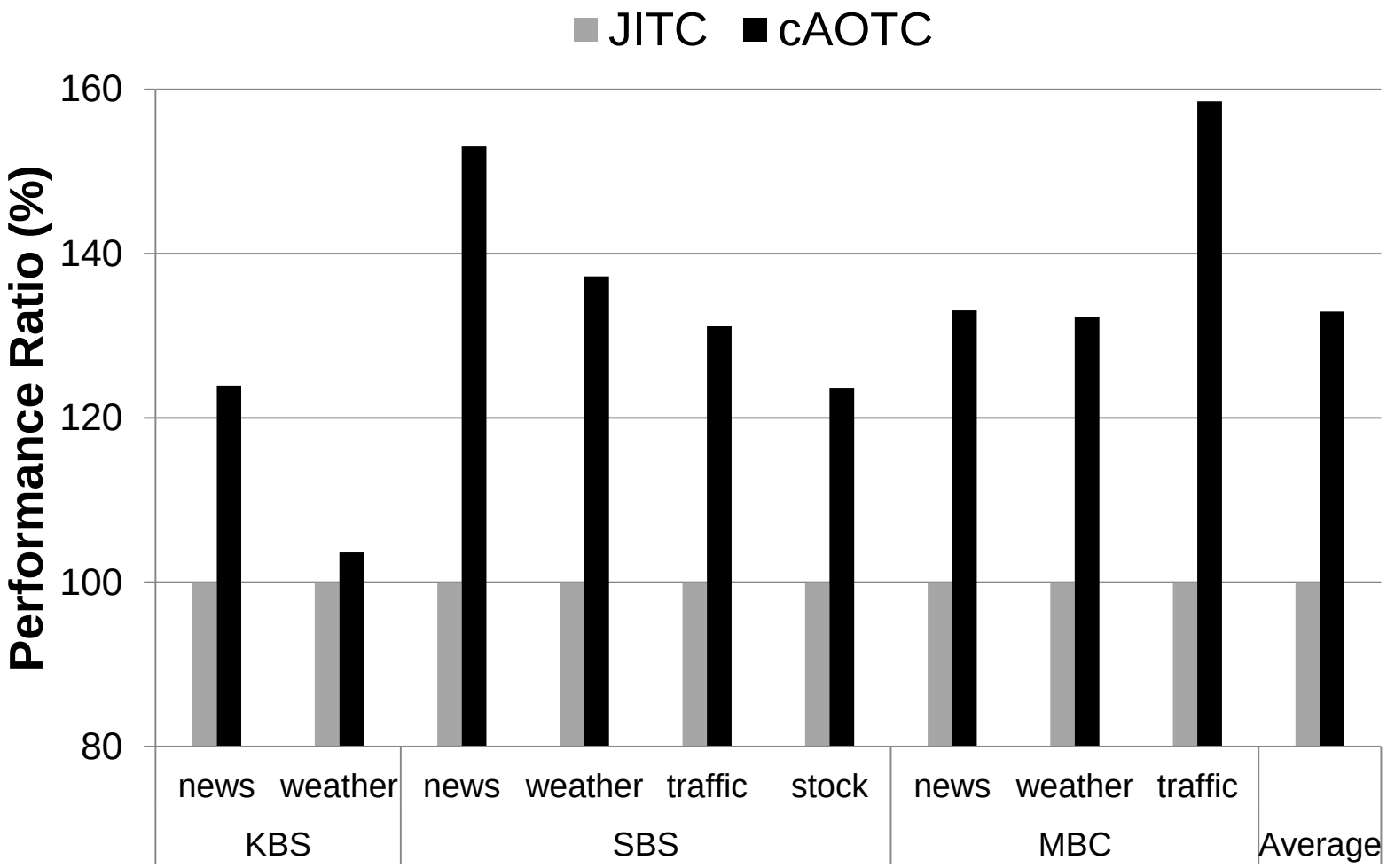


➔ c-AOTC for xlet methods is unbeneficial, so we do not use it

Experimental Results

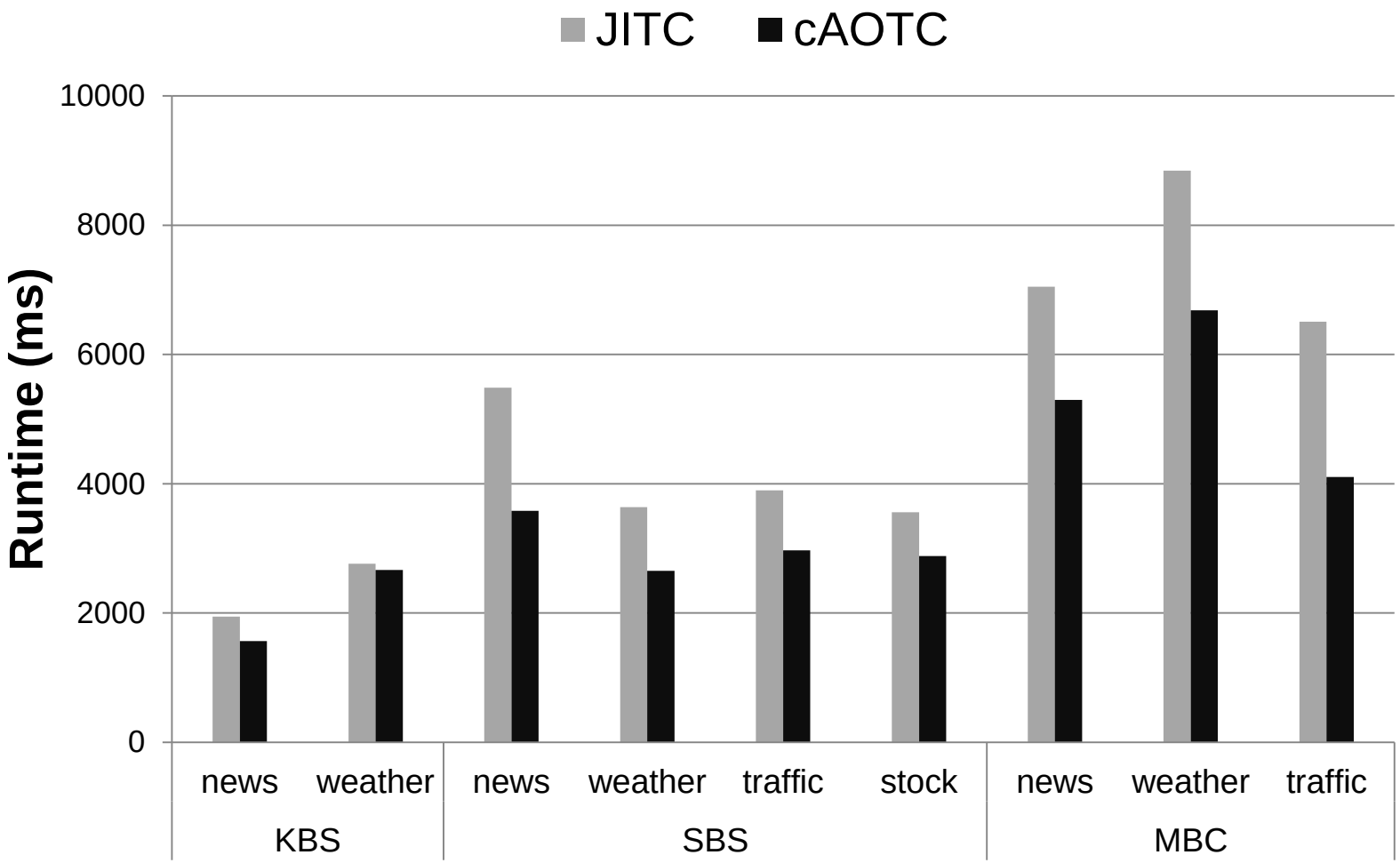
- Experimental Environment
  - DTV set-top box 333MHZ MIPS CPU with 128MB memory
  - Linux with kernel 2.6
  - Oracle's phoneME Advanced MR2 version
  - Advanced common application platform (ACAP)
  - xlets of three terrestrial TV stations in Korea
    - MBC, SBS, KBS
    - Interested in running time of each menu item

Performance Impact of c-AOTC



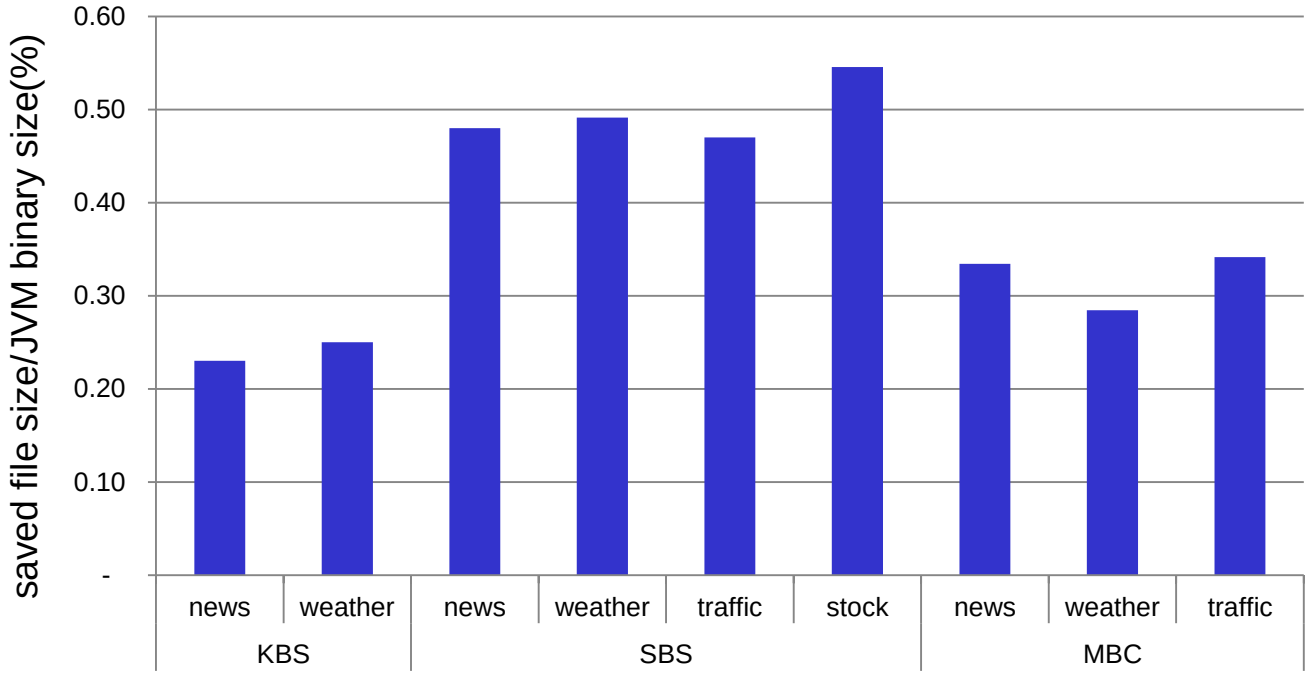
~30% performance improvement than JITC-only

Running Time of c-AOTC

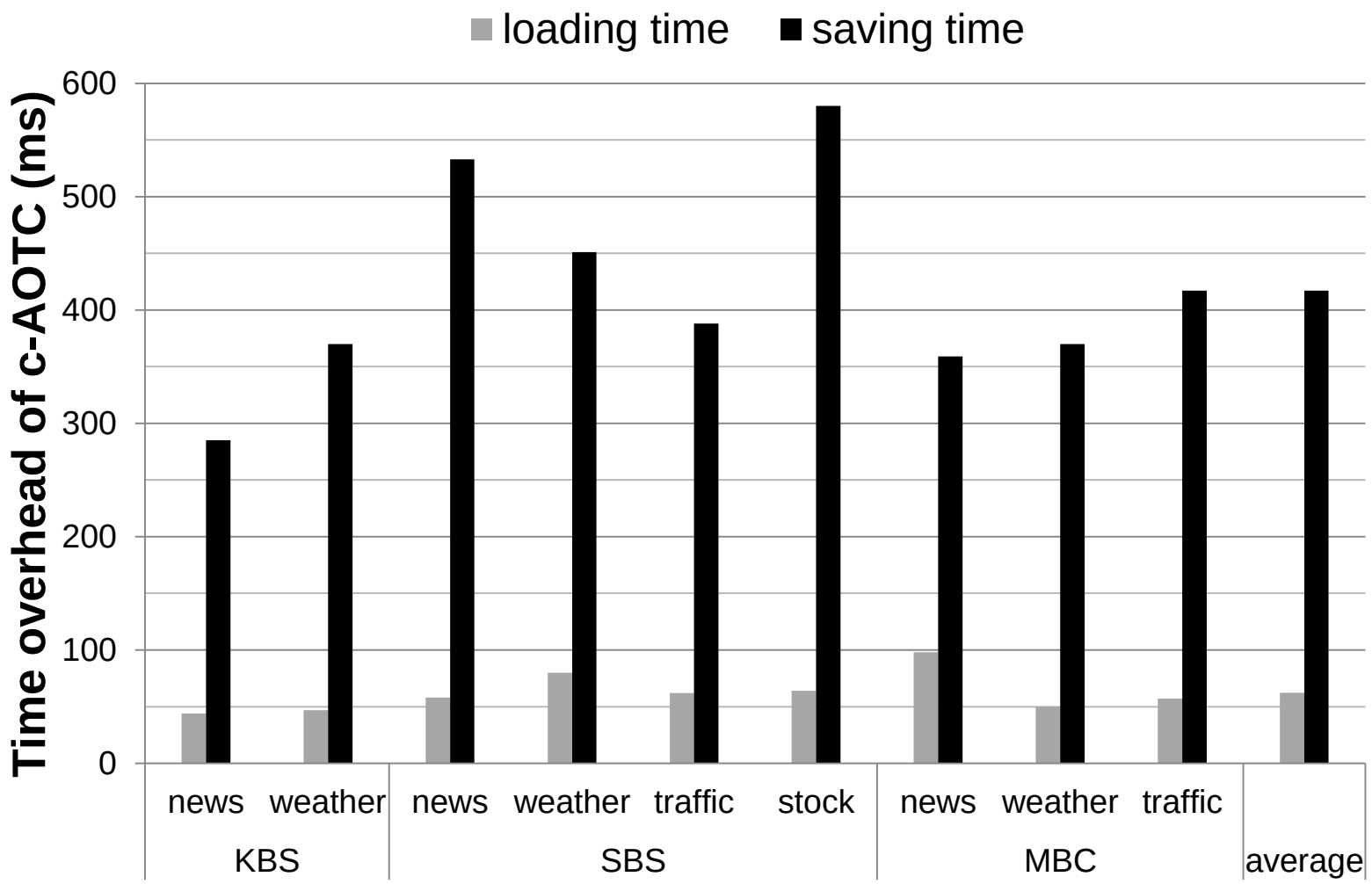


Space Overhead of c-AOTC

- The saved machine code file cause 0.5% space overhead
  - PhoneME JVM is about 70Mbyte & the saved file is under 5kb
  - PhoneME JVM include the system/middleware class and romized parts.



c-AOTC Overhead



Summary

- We employed our c-AOTC approach to commercial DTV platform and test the real xlet application

- Our c-AOTC got average 33% performance improvement

Q&A

- Please, send any question to the author.

  hongshow@snu.ac.kr