UQÀM
Université du Québec à Montréal

Sable
McGill

Porting and Tuning Inline-Threaded Interpreters

by

Etienne Gagnon

Grzegorz Prokopski

Université du Québec à Montréal

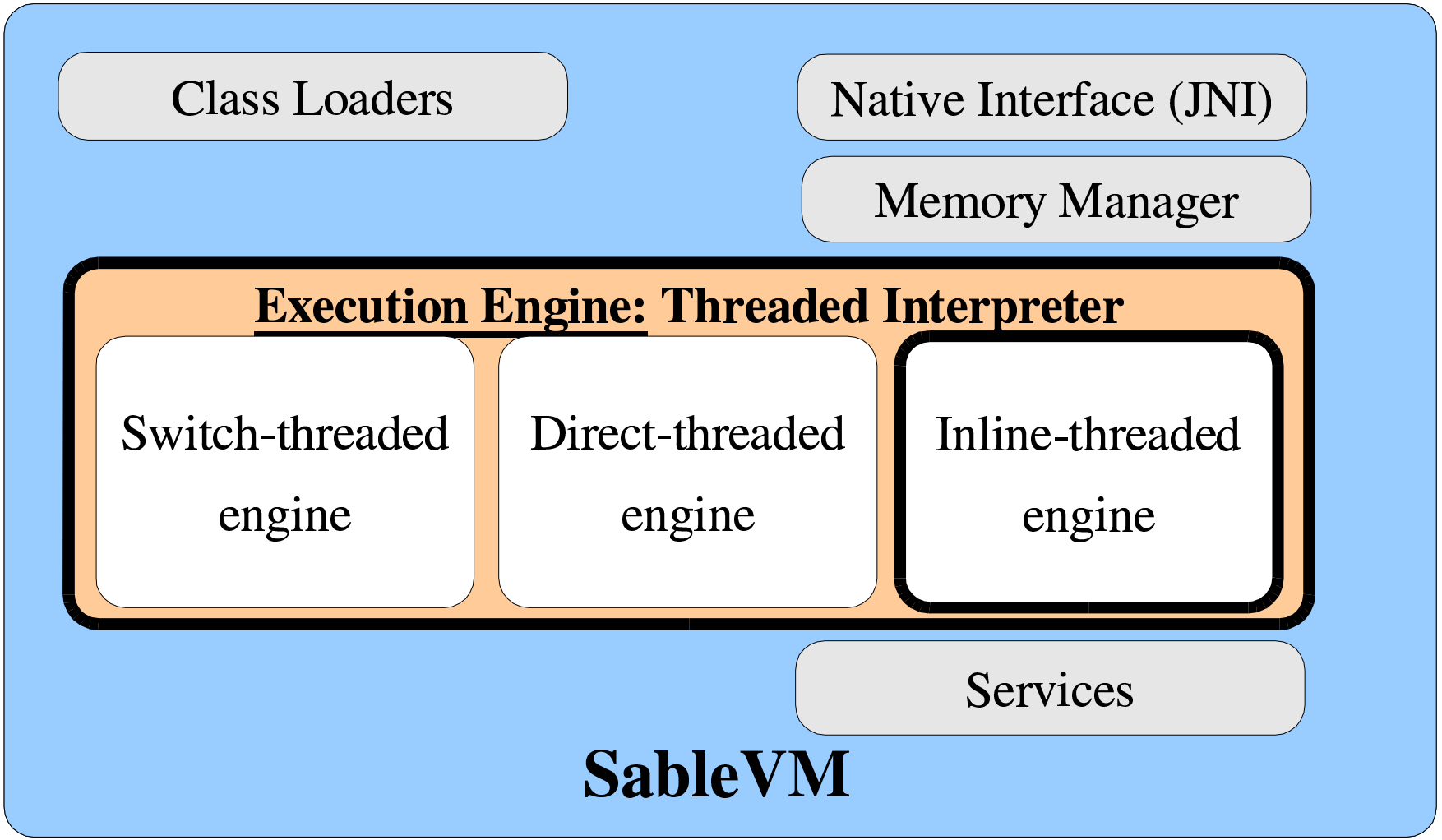
Sable Research Group

This work was partly supported by FCI, NSERC, and PAFARC(UQAM)

Outline

- Introduction to Inline-Threaded Interpretation
- SableVM Experience: Inline-Threading Challenges
- Framework for Semi-Automatic Tuning
- Conclusion

SableVM Execution Engine



UQÀM
Université du Québec à Montréal

Sable
McGill

Bytecode Interpreter

```
for (;;)
{
  bytecode = ...;
  switch(bytecode)
  {
    case INSTRUCTION1: ...;
                       break;
    case INSTRUCTION2: ...;
                       break;
...
  }
}
```

Central Dispatch

Direct-Threaded Interpreter

```
INSTRUCTION1: ...; DISPATCH;
INSTRUCTION2: ...; DISPATCH;

...
```

Distributed Dispatch

```
#define DISPATCH goto **(pc++);
```

UQÀM
Université du Québec à Montréal

Sable
McGill

Inlined-Threaded Interpreter

Dynamically computed

```
(Instruction1 body)...;
(Instruction2 body)...;
(Instruction3 body)...;
DISPATCH;
```

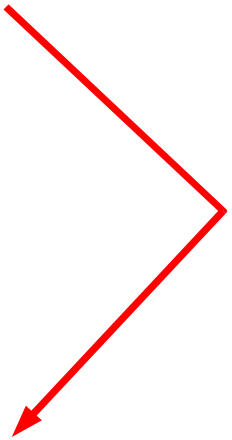
- Introduced in [PR98]
- Eliminates dispatch overhead within *basic blocks*

What can go wrong?

- Many things!
  - Two-mode instructions
    - see [CC2003] paper on *preparation sequences*
  - Relative jumps to target out of instruction body
  - Compiler optimizations
  - Compiler dubious optimisations
  - Difference between platforms / compiler versions

Relative Jumps

```
(Instruction1 body)...;
(Instruction2 body)...;
(Instruction3 body)...;
DISPATCH;
```



???

Solution: Do not inline instruction

Compiler Optimizations

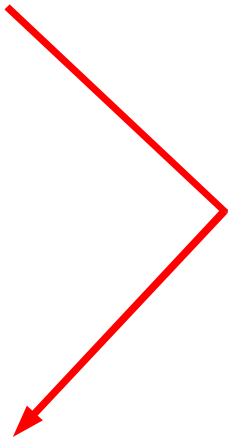
```
...head...
if (...) {
   ...then part...
}
...tail...
```

```
... head ...
   beq then_part
tail:
   ...tail...
   DISPATCH
then_part:
   ...then_part
   jump tail
```

```
...head...
if (...) {
   ...then part...
}
...tail...
```

```
   ... head ...
   beq then_part
tail:
   ...tail...
   DISPATCH
then_part:
    ...then_part
   jump tail
```

```
(Instruction1 body)...;
(Instruction2 body)...;
(Instruction3 body)...;
DISPATCH;
```



Missing then part!!

Solution: Do not inline instruction

Compiler Dubious Optimization

```
/* Actual SableVM implementation for DISPATCH */

goto *((pc++)->implementation);
```

Compiles into

???

```
goto *((pc++)->implementation);
```

Compiles to PowerPC assembly (GCC 3.3)

```
    lwz r11, 0(r27)   ;; r11 = pc->implementation
    addi r27, r27, 4  ;; pc = pc + 4
    mr r8, r11        ;; r8 = r11
    b goto_impl       ;; relative jump to goto_impl
    ...
goto_impl:
    mtctr r8          ;; ctr = r8
    bctr              ;; goto *ctr
```

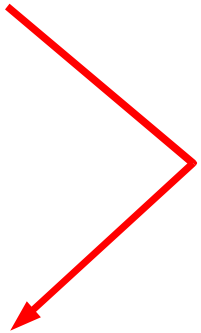
```
goto *((pc++)->implementation);
```

Compiles to PowerPC assembly (GCC 3.3)

```
lwz r11, 0(r27)   ;; r11 = pc->implementation
addi r27, r27, 4  ;; pc = pc + 4
mr r8, r11        ;; r8 = r11
b goto_impl       ;; relative jump to goto_impl
```

```
...
goto_impl:
    mtctr r8          ;; ctr = r8
    bctr              ;; goto *ctr
```

```
(Instruction1 body)...;
(Instruction2 body)...;
(Instruction3 body)...;
DISPATCH;
```



Missing goto_impl !!!

Consequence: **Inline-threading BREAKS with GCC 3.3 on PowerPC.**

UQÀM
Université du Québec à Montréal

Sable
McGill

Solution for broken DISPATCH

```
#if defined(__powerpc)
__volatile__ __asm__ {
   lwz r8, 0(r27)    ;; r8 = pc->implementation
   addi r27, r27, 4  ;; pc = pc + 4
   mtctr r8          ;; ctr = r8
   bctr              ;; goto *ctr
}
#else
  goto *((pc++)->implementation);
#endif
```

SableVM Inline-Threading Tuning Framework

- Need to specify different inlinability for each specific instruction
  - inlinability of an instruction is affected by
    - underlying platform
    - compiler version and selected options
- Need to *test/check* inlinability of each instruction
  - 345 instructions (in SableVM)!!!

Tuning Framework

- Part I : Specifying inlinability in the SableVM source code
  - We developped a set of m4 macros to avoid cluttering instruction bodies with inlinability information
  - Inlinability is specified on tables
    - row = instruction
    - column = arch&compiler-version&options

Tuning Framework

- Part II : Testing inlinability
  - We added a testing mode (./configure option) to SableVM
    - We trap signals (segmentation faults,...) and write a diagnostic to output stream
  - We developped a suite of tests written in jasmin (Java bytecode assembly)
- Manual tuning is still required, but greatly simplified

Conclusion

- Inline threaded interpretation removes dispatch overhead within basic blocks
- It requires fine-tuning for each platform and compiler/options
- SableVM implements a semi-automatic framework to check and maintain tuning information easily
- Download: **http://www.SableVM.org/**