Software Speculative Multithreading for Java

Chris Pickett    Clark Verbrugge
School of Computer Science, McGill University
{cpicke,clump}@sable.mcgill.ca

Allan Kielstra
IBM Toronto Lab
kielstra@ca.ibm.com

October 24th, 2007

ACM Student Research Competition at OOPSLA 2007
Method Level Speculation
// execute foo non-speculatively
r = foo (a, b, c);
// execute foo non-speculatively
r = foo (a, b, c);

// execute past return point speculatively in parallel
// execute foo non-speculatively
r = foo (a, b, c);

// execute past return point speculatively in parallel
if (r > 10) // predict return value
{
    ...
}

Example

// execute foo non-speculatively
r = foo (a, b, c);

// execute past return point speculatively in parallel
if (r > 10) // predict return value
{
    s = o1.f; // buffer heap reads
    o2.f = r; // buffer heap writes
}
// execute foo non-speculatively
r = foo (a, b, c);

// execute past return point speculatively in parallel
if (r > 10) // predict return value
{
  s = o1.f; // buffer heap read
  o2.f = r; // buffer heap writes
}

// invoke bar() speculatively
o3.bar();
// execute foo non-speculatively
r = foo (a, b, c);

// execute past return point speculatively in parallel

if (r > 10) // predict return value
{
    s = o1.f; // buffer heap reads
    o2.f = r; // buffer heap writes
}

// invoke bar() speculatively
o3.bar();

// stop speculation
synchronized (o4) { ... }
Experimental Analysis

System configuration:

- SableVM Java bytecode interpreter
- 4-way 1.8 GHz Opteron (circa 2004)
- create child task on every invocation
- out-of-order speculation: yes
- in-order speculation: no

Metric: speedup vs. sequential execution

- control: speculate but always force abort
- treatment: speculate and allow commit
Implementation

SableVM (McGill)
  research
  Java bytecode interpreter
Implementation

SableSpmt

SableVM backend
research
fixed-point
interpreter

2003 - 2005
Implementation

SableSpmt

SableVM project

research

J9 (IBM)

production

JVM with

JIT compiler

2003 - 2005

interpreter
Implementation

SableSpMT

SableVM (McGill)
- research
- Java bytecode interpreter
- 2003 - 2005

libspmt
- VM and language independent speculation library
- 2006 - 2007

J9 (IBM)
- production JVM with JIT compiler
- 2007+


Immediate Future Work

Reduce overhead based on initial system profiling:

- Adaptive return value prediction
  - Hypothesis: there is an ideal predictor for each callsite

- In-order (or “nested”) method level speculation
  - Hypothesis: if children create children it exposes more parallelism

- Online fork heuristics
  - Hypothesis: online profiling reveals the best places to create threads

All of these things are presently implemented in libspmt.

- Need to connect J9 + JIT to libspmt
- Need to develop controlled experimental analyses
Related Work

Hardware findings:
- Method speculation subsumes loop speculation (Chen, PACT’98)
- Method speculation is viable for Java in hardware (Chen, ISCA’03)
- Return value prediction helps Java method speculation (Hu, JILP’03)
- Fork heuristics help Java method speculation (Whaley, ICPP’05)

Software findings:
- Loop speculation is viable in software (Cintra, TPDS’05)
- Speculative (or “safe”) futures work for Java in software (Welc, OOPSLA’05)
- Coarse-grained speculation is viable in software (Ding, PLDI’07)
- Method speculation works for Haskell in software (Harris, ICFP’07)

Our work is unique because it:
- is software-based
- is JVM-based
- handles SPEC JVM98