INSanE
Jam, Oliveira, Petit
NSan
Nsan Instrumentation
Contributions
NSan Adaptation
Opaque Shadows
Instrumentation
Runtime examples
DoublePrec
MCASync
Results - Work in Progress
CPU2006
References

INSanE - Interface for Numerical Stability Sanitizer Extension

Mathys Jam[1] Pablo Oliveira[1] Eric Petit[2]

[1]Université Paris-Saclay, UVSQ, Li-PaRAD

[2]Intel Corporation

2021

Introduction to NSan

INSanE
Jam, Oliveira, Petit
NSan
Nsan Instrumentation
Contributions
NSan Adaptation
Opaque Shadows Instrumentation
Runtime examples
DoublePrec
MCASync
Results - Work in Progress
CPU2006
References

- Numerical Stability Sanitizer, by C.Courbet from Google[1]
- An llvm compiler pass and associated shadow memory runtime
- Purpose build optimized IR instrumentation

This presentation is about INSanE, which is an extension of Nsan.

[1] Clement Courbet. "NSan: A Floating-Point Numerical Sanitizer". In: *Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction*. 2021, 83–93

Instrumentation

INSanE
Jam, Oliveira, Petit
NSan
Nsan Instrumentation
Contributions
NSan Adaptation
Opaque Shadows
Instrumentation
Runtime examples
DoublePrec
MCASync
Results - Work in Progress
CPU2006
References

- Associates every FP values **with an extended precision** shadow value 'S'

$$x \rightarrow S(x)$$

- Replicate every FP operation in the shadow space

$$x \circ y \rightarrow S(x) \circ S(y)$$

- Error checking and shadow handling is done in a separate runtime
- Similar to FPDebug[2]

[2] Florian Benz, Andreas Hildebrandt, and Sebastian Hack. "A dynamic program analysis to find floating-point accuracy problems". In: *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '12, Beijing, China - June 11 - 16, 2012*. ACM, 2012, pp. 453–462

Instrumentation Example

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

Original IR :

```
%b = fmul float 2.0, float 2.0
store float %b, float* %a.ptr
```

After Instrumentation :

```
%b = fmul float 2.0, float 2.0
%b.shadow = fmul double 2.0, double 2.0
store float %b, float* %a.ptr
%check.result = call i32 @__nsan_check_float(float %b, double %b.shadow)
; ...  Maybe report an error

%a.shadow.ptr = call i8* @__nsan_get_shadowptr_for_float_store(%a.ptr, 1)
%shadow.address = bitcast i8* %a.shadow.ptr, double*
store double %b.shadow, double* %shadow.address
```

Additional runtime and instrumentation

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

Nsan also performs special instrumentation

- Math intrinsics are widened when their extended counterparts are known
- FP comparisons and branching
- Intercepting all memory operation (reuse of LLVM sanitizer tools): L/S, memcpy, memset...
- Split the memory space in 4 regions, 1 for app, 2x1 for shadow memory, 1 for type checking.
- Translating an app address to the shadow space is just a simple afine function.
- Type checking and tracking aliased store to memory
  $\rightarrow$ This is a key ingredient in the secret sauce!

Pros/Cons

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

Pros :

- May be merged into LLVM's trunk meaning it will be generally available
  - LLVM license
  - Already in Upstream candidate process
- Fast approach, accurate enough for most users
- Shadow memory handling system

Pros/Cons

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

Pros :

- May be merged into LLVM's trunk meaning it will be generally available
  - LLVM license
  - Already in Upstream candidate process
- Fast approach, accurate enough for most users
- Shadow memory handling system

Cons :

- Does not detect every instability, or wrongly report them.
- Target only debugging, not validation and optimization.
- Repurposing the tool requires deep rewriting of the pass and runtime.
- Tied to LLVM

Contributions

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

INSanE : Interface for Numerical Sanitizer Extension

- Extend NSan instrumentation with a generic interface
- Provide a framework for easily tunables runtimes implemented in C++
- Runtimes can focus on analysis while NSan provides the instrumentation and shadow handling
- Two examples of runtimes : DoublePrec, MCASynchrone

Workflow

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

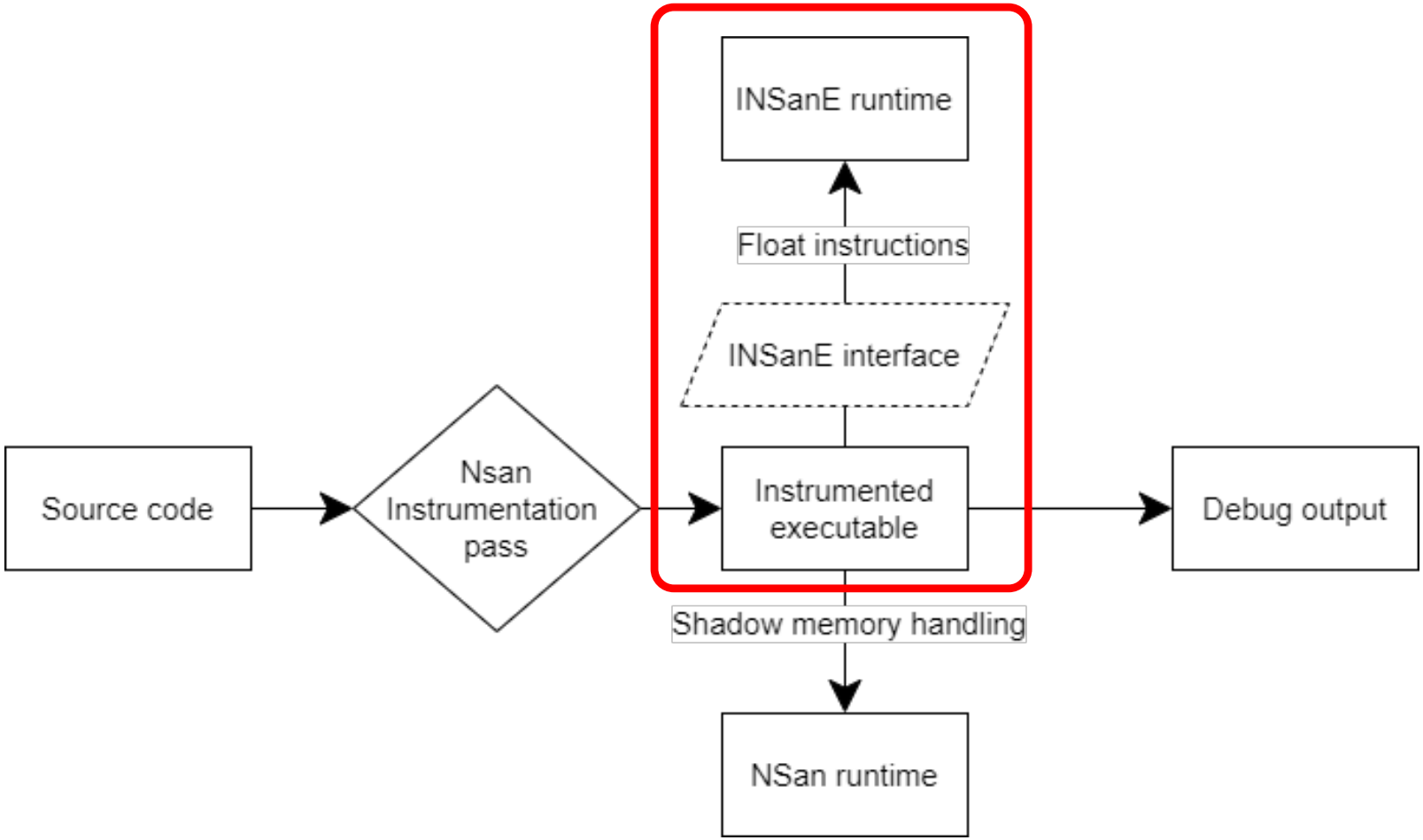
DoublePrec

MCASync

Results - Work in Progress

CPU2006

References



Shadow scale

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

New analysis implies custom shadow types

- What is inside the shadows ? Where do we define it ?
- Is 64/128 bits sufficient for a shadow ?

NSan's shadows are not big enough for all runtimes. We introduce an "extended memory' mode, with x4 shadows

FPType	x2 shadow size	x4 shadow
Float	64	128
Double	128	256

Opaque Shadows

Opaque Shadows

The instrumentation and the interface only knows the shadow's size : their content is defined in the runtime

Example (x4 shadow) :

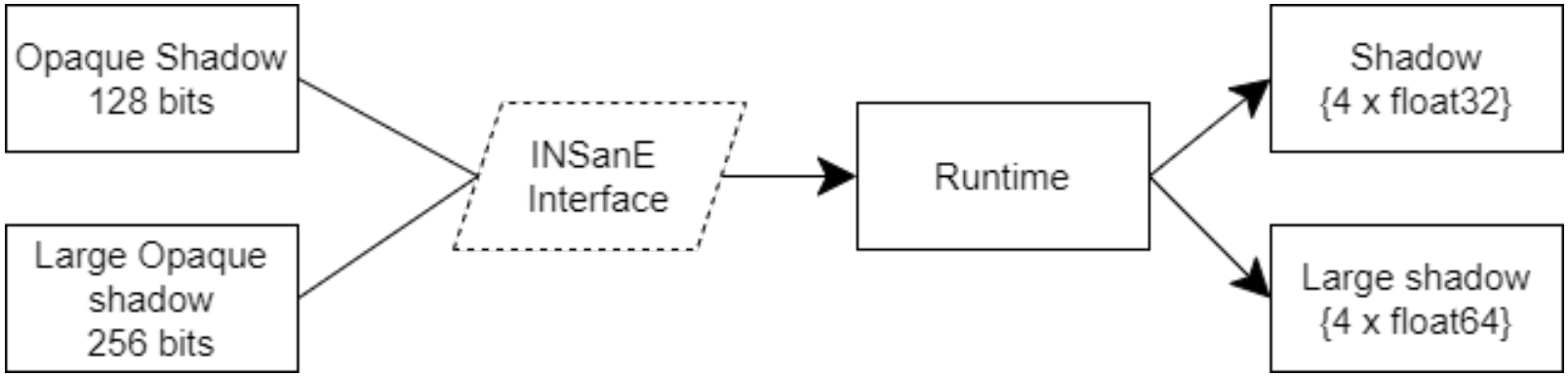


Figure: Examples of shadows as used in the MCASync runtime

Instrumentation mode

INSanE
Jam, Oliveira, Petit
NSan
Nsan Instrumentation
Contributions
NSan Adaptation
Opaque Shadows
Instrumentation
Runtime examples
DoublePrec
MCASync
Results - Work in Progress
CPU2006
References

NSan mode versus INSanE mode

- Shadow handling is preserved
- Most of the sructure of the Instrumentation remains the same
- FP Instructions are replaced by call to INSanE's interface
- NSan runtime calls are replaced by INSanE's interface calls

From a user perspective:

The instrumention mode mostly affect which runtime (NSan / INSanE) is called

Interface definition

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

The interface includes :

- Binary / Unary operators (+, -, /, *, neg)

```
float __insane_float_fadd(float a, OpaqueShadow* sa,
                          float b, OpaqueShadow* sb, OpaqueShadow* res);
```

- Comparisons $>, \geq, <, \leq, =, \neq$, ordered / unordered Type conversion cast operations
- Shadow constructor

Issues with vector ABI

INSanE
Jam, Oliveira, Petit
NSan
Nsan Instrumentation
Contributions
NSan Adaptation
Opaque Shadows
Instrumentation
Runtime examples
DoublePrec
MCASync
Results - Work in Progress
CPU2006
References

```
float __insane_float_fadd(float a, OpaqueShadow* sa,
                          float b, OpaqueShadow* sb, OpaqueShadow* res);

void __insane_double_v4_make_shadow(<4 x double> a, LargeShadow** sa);
```

This interface raises some issues :

- Opaque shadows are structure of size $\geq$64 bits
- We cannot build vectors of structures
- Alignment issues
- ...

Issues

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

Solutions

- Every shadow is handled as a pointer.
- In the instrumentation, we use vectors of pointers.
- Pointer vectors are translated into array of pointers in C/C++.
- The runtime and the instrumented code must use the same compilation flags

Proof of concepts

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

We have two working runtimes :

- DoublePrec : replicates NSan behaviour, used for benchmarking and debugging.
- MCASync : Synchronous stochastic runtime

DoublePrec

Implementation example for fsub overload in DoublePrec

```
template <typename MetaFloat>
typename MetaFloat::FPType InsaneRuntime<MetaFloat>::Sub(
    FPType LeftOperand, ShadowType **LeftShadowOperand,
    FPType RightOperand, ShadowType **const RightShadowOperand,
    ShadowType **Res) {

  using DoublePrecShadowType = DoubleprecShadowFor<ShadowType>;
  // We align both operands
  DoublePrecShadowType LeftShadow[VectorSize], RightShadow[VectorSize];
  CopyAndAlign<VectorSize>(LeftShadow, LeftShadowOperand);
  CopyAndAlign<VectorSize>(RightShadow, RightShadowOperand);

  auto ResShadow = reinterpret_cast<DoubleprecShadowFor<ShadowType> **>(Res);

  for (int I = 0; I < VectorSize; I++)
    ResShadow[I]->val = LeftShadow[I].val - RightShadow[I].val;

  return LeftOperand - RightOperand;
}
```

MCASync

Synchronous Monte-Carlo Arithmetic :

- Associate every IEEE FP Value with 3 orbitals [4]
- We introduce a small rounding error in each orbital during computations using MCA [3, 5]
- Using those orbitals we can approximate the number of significant digits
- We keep the IEEE value to follow the same path as an uninstrumented run

Performance results

INSanE
Jam, Oliveira, Petit
NSan
Nsan Instrumentation
Contributions
NSan Adaptation
Opaque Shadows
Instrumentation
Runtime examples
DoublePrec
MCASync
Results - Work in Progress
CPU2006
References

Bench	Original	NSan	Doubleprec	Slowdown vs NSan
lbm	1.55s	127s	163s	1.28
milc	3.04s	633s	877s	1.38
namd	6.91s	658s	941s	1.43
dealII	7.72s	455s	682s	1.49
soplex	0.0116s	0.220s	0.311s	1.41
povray	0.297s	16.2s	27.4s	1.69
sphinx3	0.814s	25.6s	28s	1.09
Average slowdown vs Original	1	78.45	110.22	
Average				**1.4**

Figure: Spec2006 benchmarks : Doubleprec runtime performance against NSan

Conclusion

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

- Work in progress but very encouraging results
- The addition of compiler support for shadow memory is opening a large set of new opportunities
- There is probably no technical difficulty to merge Verificarlo and Nsan in a single compiler pass

References I

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

Florian Benz, Andreas Hildebrandt, and Sebastian Hack. "A dynamic program analysis to find floating-point accuracy problems". In: *ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI '12, Beijing, China - June 11 - 16, 2012*. ACM, 2012, pp. 453–462.

Clement Courbet. "NSan: A Floating-Point Numerical Sanitizer". In: *Proceedings of the 30th ACM SIGPLAN International Conference on Compiler Construction*. 2021, 83–93.

References II

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

Christophe Denis, Pablo de Oliveira Castro, and Eric Petit. "Verificarlo: Checking Floating Point Accuracy through Monte Carlo Arithmetic". In: *23nd IEEE Symposium on Computer Arithmetic, ARITH 2016, Silicon Valley, CA, USA, July 10-13, 2016*. 2016, pp. 55–62. DOI: 10.1109/ARITH.2016.31. URL: http://dx.doi.org/10.1109/ARITH.2016.31.

Fabienne Jézéquel and Jean-Marie Chesneaux. "CADNA: a library for estimating round-off error propagation". In: *Computer Physics Communications* 178.12 (June 2008), pp. 933–955. DOI: 10.1016/j.cpc.2008.02.003. URL: https://hal.archives-ouvertes.fr/hal-01146486.

References III

INSanE

Jam, Oliveira, Petit

NSan

Nsan Instrumentation

Contributions

NSan Adaptation

Opaque Shadows

Instrumentation

Runtime examples

DoublePrec

MCASync

Results - Work in Progress

CPU2006

References

Devan Sohier et al. "Confidence Intervals for Stochastic Arithmetic". In: (2021). DOI: arXiv:1807.09655v3. URL: https://arxiv.org/abs/1807.09655.