



# Memory Checker Technology Preview *(and how we're building it)*

Jasper Kamperman, PhD  
Product Manager  
Developer Products Division

Gelato\* ICE, April 25, 2006

Gelato ICE 2006. Copyright © 2006, Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.



# Disclaimer



- Information in this document is provided in connection with Intel® products. No license, express or implied, by estoppels or otherwise, to any intellectual property rights is granted by this document, and Intel® makes no representations as to the necessity of obtaining licenses from any third party. Except as provided in the Terms and Conditions of Sale for such products, Intel® assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel® products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other Intel® intellectual property right. Intel products are not intended for use in medical, life saving, or life sustaining applications.
- The information in this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel®. Intel assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.
- Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel®.
- The Intel® product may contain design defects or errors known as errata which may cause the product to deviate from published specifications.

Gelato ICE 2006. Copyright © 2006, Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.



# Agenda



- James Reinders' Keynote Recap
- Memory Checker Technology Preview
- How we're building it
  - Overview of the Pin system for Dynamic Binary Instrumentation
  - Memory Checker Internals
- Call to action



# James Reinders' Keynote Recap



- We've heard your requests for a memory checker for Itanium® processors
- You're invited to join the VTune™ Performance Analyzer Beta that *includes a Technology Preview of our Memory Checker*
- We're providing CD-ROMs with this beta release at the Intel booth

Get the beta at the Intel booth, or email  
[vtune\\_beta@intel.com](mailto:vtune_beta@intel.com)



# Memory Checker Technology Preview



- First instruction-level memory checker for Intel® Itanium® 2 processors
- Finds root cause of memory leaks & memory corruption
- Pinpoints the exact instruction
- Shows relevant stacks (allocation, de-allocation, error)
- Use it in production environments
  - No need to re-compile, re-build, re-link, or manage multiple versions of shared objects
  - Compiler-independent (C, C++ and Fortran)
  - Works on optimized code
  - Produces valuable information even if you compiled without debugging information (-g)
  - Just prepend "memchkr" to your command line and go!
- Scales on multi-core/multi-processor systems



# Memory Leaks & Corruption

```
1. void doitx()
2. {
3.     char *p = (char *)malloc(4);
4.     p[4]='a';
5.     char c = p[5];
6.     c = p[0];
7.     free(p);
8.     c = p[2];
9. }
```

Wild Store

Wild Load

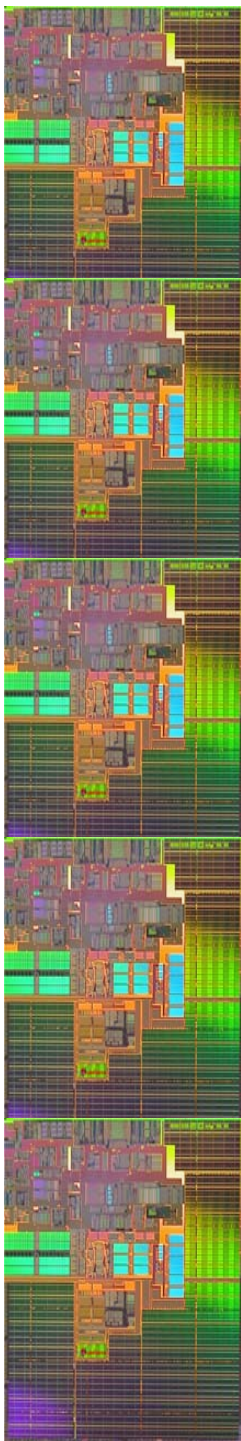
Un-initialized Load

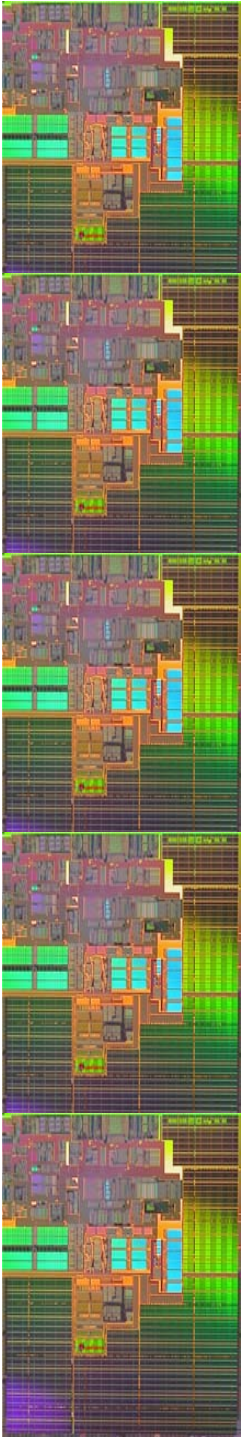
Accessing freed  
Memory



# How We're Building it, Part I

## *Overview of the Pin system for Dynamic Binary Instrumentation*

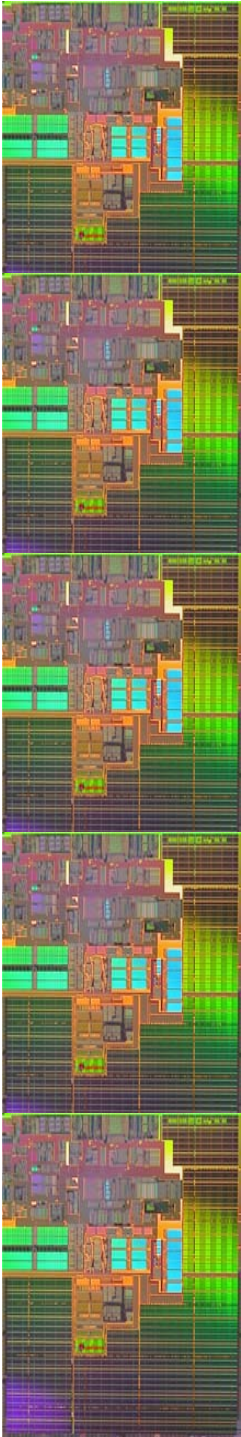




# Instrumentation



- A technique that inserts extra code into a program to collect runtime information
- Instrumentation approaches:
  - Source instrumentation:
    - Instrument source code
  - **Binary instrumentation:** ← Pin approach
    - Instrument executables directly

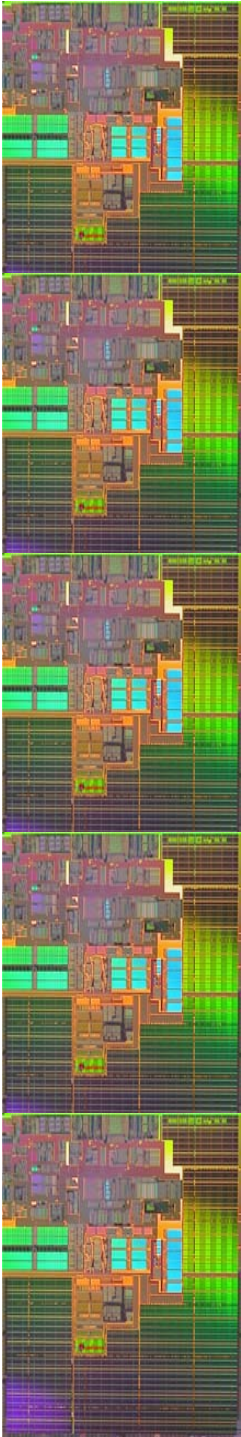


# Pin Overview



- A system for binary instrumentation
- Provides rich APIs to write instrumentation tools in C/C++ (called Pintools)
- Supports IA-32 including Intel® EM64T, Itanium® and Intel® StrongARM\* microarchitectures (not Memory Checker)
- Available free of charge (no source, no support)
  - Supports Linux\* and Mac OS\* (not Memory Checker)
- Dynamic instrumentation
  - Instruments a program while it is running

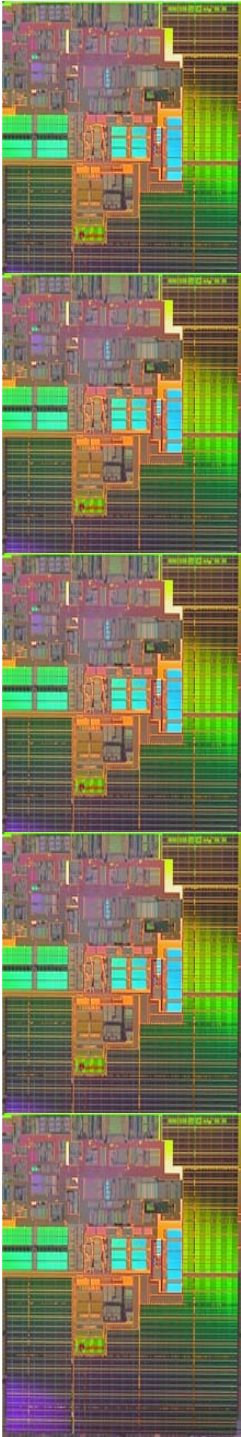
**Pin is a system for building instrumentation tools on Intel® platforms**



# Advantages of Pin (1)



- Resulting tools are easy to use because of dynamic instrumentation, don't need:
  - Source code
  - Recompilation
  - Post-linking effort
- Cross-platform support
  - Most Pintools are source compatible on IA-32, Intel® EM64T, Itanium® 2 and Intel® StrongARM\* architectures
  - Support multiple Operating Systems:
    - Linux\*
    - Mac OS\* (not Memory Checker)



# Advantages of Pin (2)

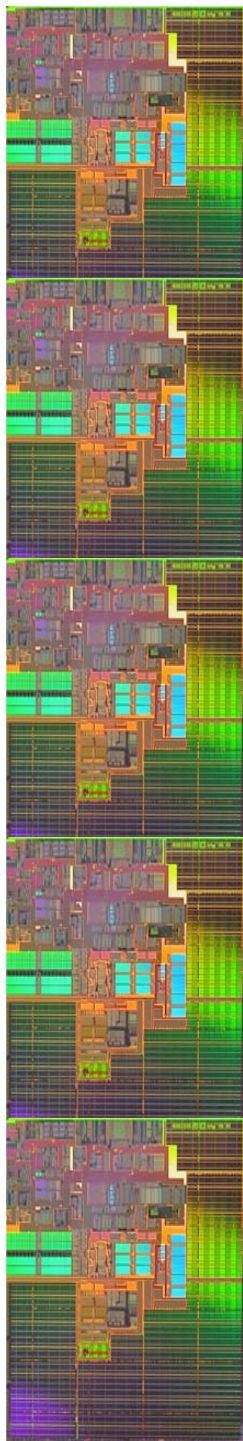


- Robust on real-life applications
  - Pin has been used on databases, search engines and many HPC applications
- Preserves original program behavior
- Efficient
  - Pin automatically optimizes instrumentation code
  - Pin can attach instrumentation to an already running process (not Memory Checker)



# Pin Concepts

- **Instrumentation routines** define where instrumentation is **inserted**
  - e.g. before instruction
    - ☞ **Occurs at compile time (JIT time)**
- **Analysis routines** define what to do when instrumentation is **activated**
  - e.g. increment counter
    - ☞ **Occurs at runtime**



# Instrumentation Points



Instrument points are relative to an **instruction**:

– *Before*  
(*IPOINT\_BEFORE*)

– After:

- Fall-through edge  
(*IPOINT\_AFTER*)
- Taken edge  
(*IPOINT\_TAKEN*)

```
cmp.eq p9,p8=0,r15
count()
(p9)br.cond.few <L1>
count()
adds r18=1,r0
<L1>:
count()
adds r18=8,r0
```



## Arguments to Analysis Routines

- **IARG\_INST\_PTR**
  - Instruction pointer (program counter) value
- **IARG\_UINT32 <value>**
  - An integer value
- **IARG\_REG\_VALUE <register name>**
  - Value of the register specified
- **IARG\_BRANCH\_TARGET\_ADDR**
  - Target address of the branch instrumented
- **IARG\_MEMORY\_READ\_EA**
  - Effective address of a memory read
- And many more ... (refer to the Pin manual for details)

# Example



## Instrumentation to count instructions

```
counter++;  
sub r15=24,r17  
counter++;  
cmp.eq p9,p8=0,r15  
counter++;  
(p9)br.cond.few <L1>  
counter++;  
adds r18=1,r0  
counter++;  
L1 :add r18=8,r0
```

# Instruction Count Output



```
$ /bin/ls
```

```
Makefile atrace.o imageload.out itrace  
proccount Makefile.example imageload  
inscount0 itrace.o proccount.o atrace  
imageload.o inscount0.o itrace.out
```

```
$ pin -t inscount0 -- /bin/ls
```

```
Makefile atrace.o imageload.out itrace  
proccount Makefile.example imageload  
inscount0 itrace.o proccount.o atrace  
imageload.o inscount0.o itrace.out
```

```
Count 422838
```

# Example Source Code



```
#include <iostream>
#include "pin.h"
```

```
UINT64 icount = 0;
```

```
void docount() { icount++; }
```

*analysis routine*

```
void Instruction(INS ins, void *v)
```

*instrumentation routine*

```
{
    INS_InsertCall(ins, IPOINT_BEFORE, (AFUNPTR)docount,
    IARG_END);
}
```

```
void Fini(INT32 code, void *v)
```

```
{ std::cerr << "Count " << icount << endl; }
```

- Same source code works on 4 architectures
- Pin automatically and efficiently saves/restores application state

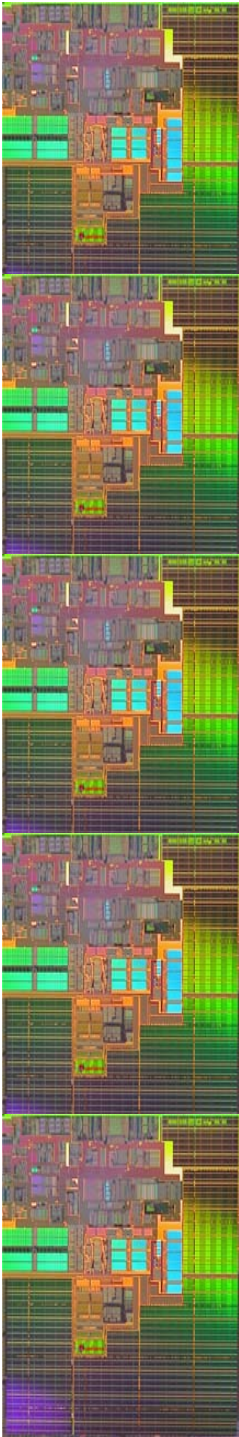
```
PIN_StartProgram();
```

```
return 0;
```

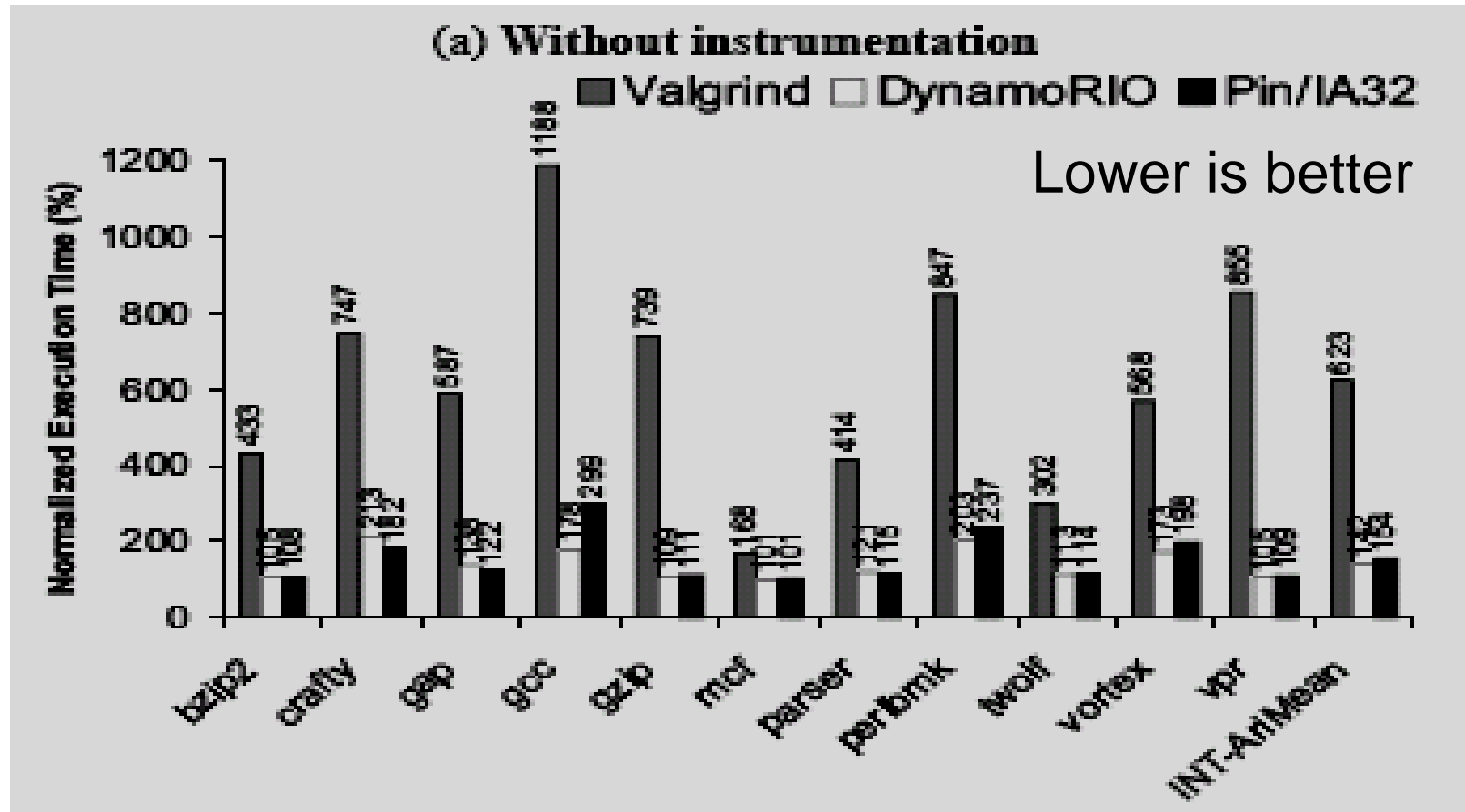
```
}
```

Gelato ICE 2006. Copyright © 2006, Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.



# Pin Performance (1)

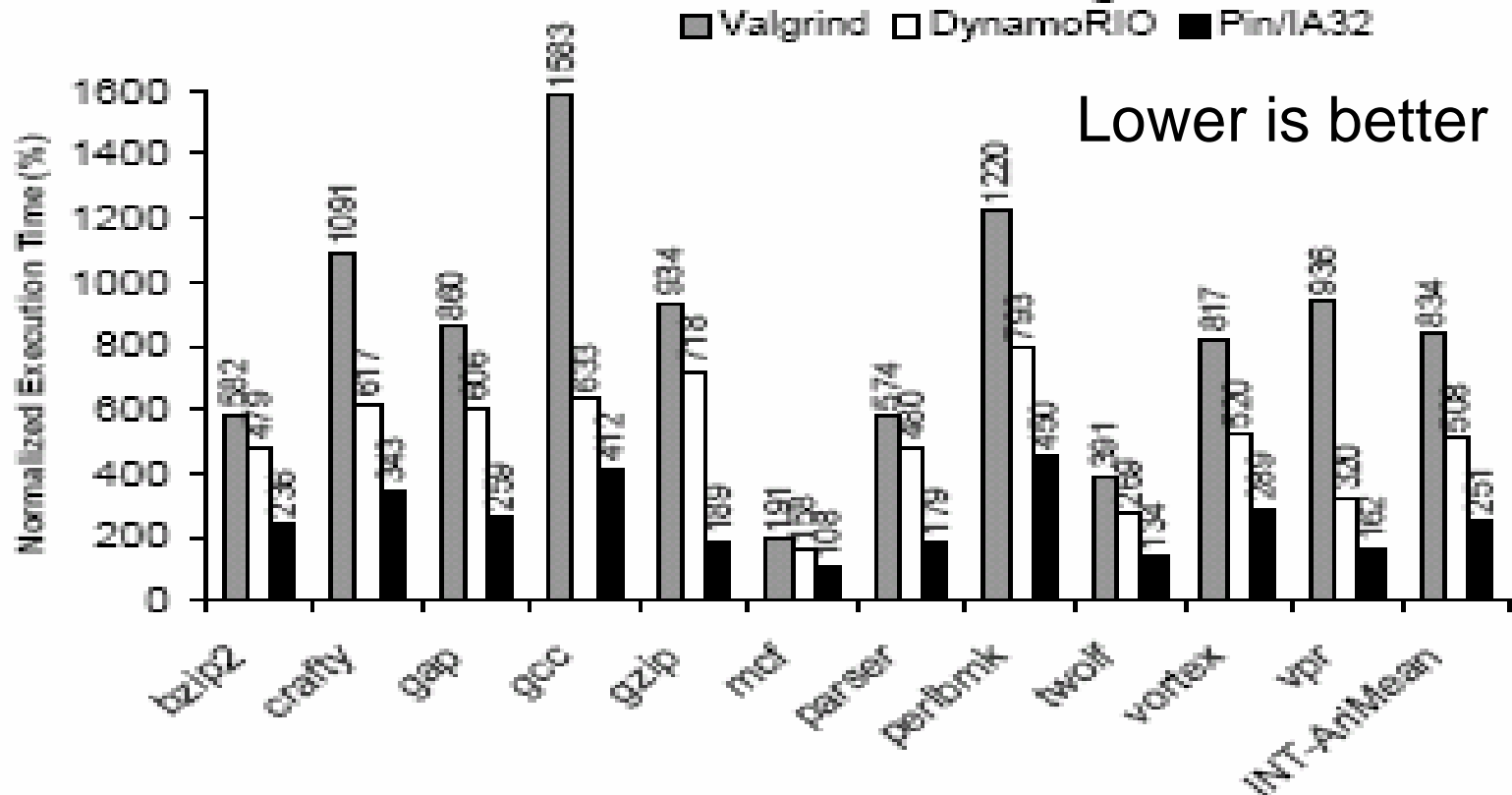


- Numbers from: *Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation*, PLDI (Programming Language Design and Implementation) June 12-15, 2005.
- Versions: Valgrind 2.2.0, DynamoRIO 0.9.3

# Pin Performance (2)



(b) With basic-block counting



- Numbers from: *Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation*, PLDI (Programming Language Design and Implementation) June 12-15, 2005.
- Versions: Valgrind 2.2.0, DynamoRIO 0.9.3

Gelato ICE 2006. Copyright © 2006, Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.



# How We're Building it, Part II

## *Memory Checker Internals*



Gelato ICE 2006. Copyright © 2006, Intel Corporation. All rights reserved.  
\*Other names and brands may be claimed as the property of others.

# Memory Checker Instruments Loads and Stores



**MEM\_LD(...);**

ld8 r9=[r10]

cmp.eq p4,p3=0,r6

(p4) br.cond L1

**MEM\_ST(...);**

st8 [r10]=r12

cmp.ne p6, p5=0,r7

(p5) br.cond L2

## Instrumentation:

Insert call to MEM\_LD()/MEM\_ST()  
before every load/store memory  
instruction

## Analysis:

- Check validity of address & memory status (allocated, freed, initialized?)
- Flag errors
- Update memory state





# Optimizations

Many optimizations are needed to reduce false positives, improve performance and make it scale, including:

- Use of inlining API to minimize analysis function calls made to track memory accesses.
- Efficient algorithm to track the call stack of application while recognizing special cases such as setjmp/longjmp.
- Dealing with partial loads and stores.
- Intercept malloc/free/new/delete calls at the right level.
- Correctly identify and flag speculative and predicated execution in Itanium® 2 processors.





# Conclusion

- First instrumentation-based memory checker supporting Intel® Itanium® 2 processors
  - Finds root cause of memory leaks and memory corruption
  - Pinpoints the exact instruction
  - Shows relevant stacks (allocation, de-allocation, error)
- Use it in production environments
  - No need to re-compile, re-build, re-link, or manage multiple versions of shared objects
  - Compiler-independent (C, C++ and Fortran)
  - Works on optimized code
  - Produces valuable information even if you compiled without debugging information (-g)
  - Just prepend "memchkr" to your command line and go!
- Scales on multi-core/multi-processor systems
- Based on the Pin system for building dynamic binary instrumentation tools

Gelato ICE 2006. Copyright © 2006, Intel Corporation. All rights reserved.

\*Other names and brands may be claimed as the property of others.





# Call to Action

- Visit the Intel booth to:
  - See a demo of the Memory Checker
  - Get a CD-ROM of VTune™ Performance Analyzer Beta Release, which includes our Memory Checker
- Signup / more info: Send mail to [vtune\\_beta@intel.com](mailto:vtune_beta@intel.com)
- To build your own tools, download Pin from <http://rogue.colorado.edu/pin>

