Adding Parallelism with Intel® Parallel Studio: No Parallelism Experience Required

Mark W. Davis
Developer Products Division
April 22, 2010

Agenda

- Overview
- Find where parallelism can be usefully added to a program
- Fix the parts of the program that prevent parallelism
- Add parallelism
- Test the revised program for correctness and performance
- Don’t panic

Agenda

- Overview
- Find where parallelism can be usefully added to a program
- Fix the parts of the program that prevent parallelism
- Add parallelism
- Test the revised program for correctness and performance
- Don’t panic

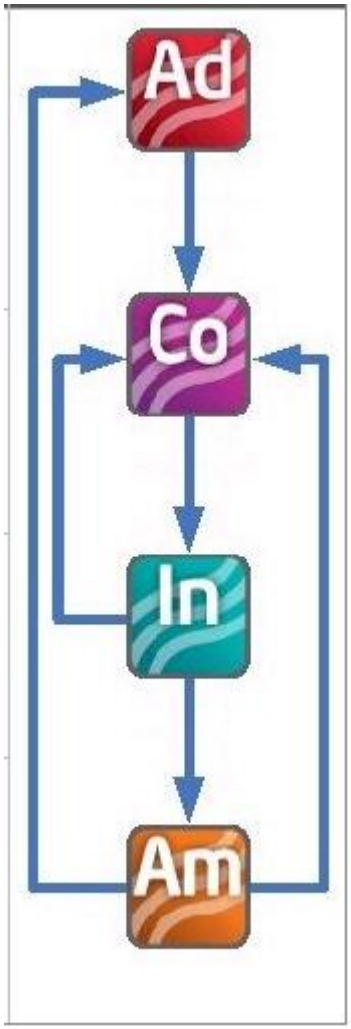
Overview – adding parallelism

- What is parallelism?
  - Overlapping portions of a serial algorithm, so it runs faster
    - Eg: Search both sides of a tree at the same time
- Start with a serial algorithm
- Modify it to be a parallel-ready serial algorithm
- Add code to
  - Perform statements in parallel
  - Synchronize the sharing of common resources

Analysis, preparation and testing of the serial code is key to simplifying adding parallelism

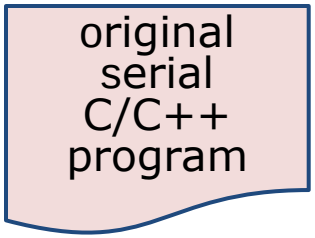
Overview – the tools

- Decide where to add the parallelism
  - Analyze the serial program
  - Prepare it for parallelism
  - Test the preparations
- Add the parallelism
  - Cilk, Intel® Threading Building Blocks, OpenMP*, o/s threads
- Find logic problems
  - Only fails sometimes
  - Place of failure changes
- Find performance problems

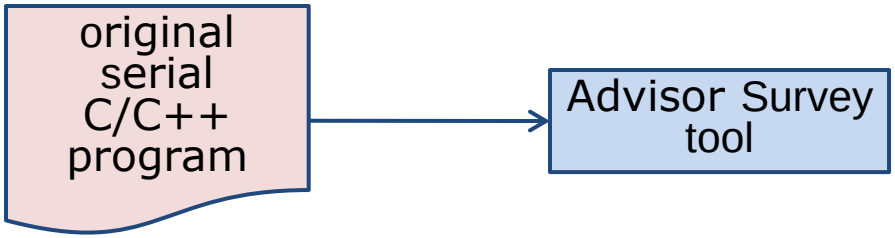


Adding and using parallelism requires enhanced tools

Overview – Intel Parallel Advisor

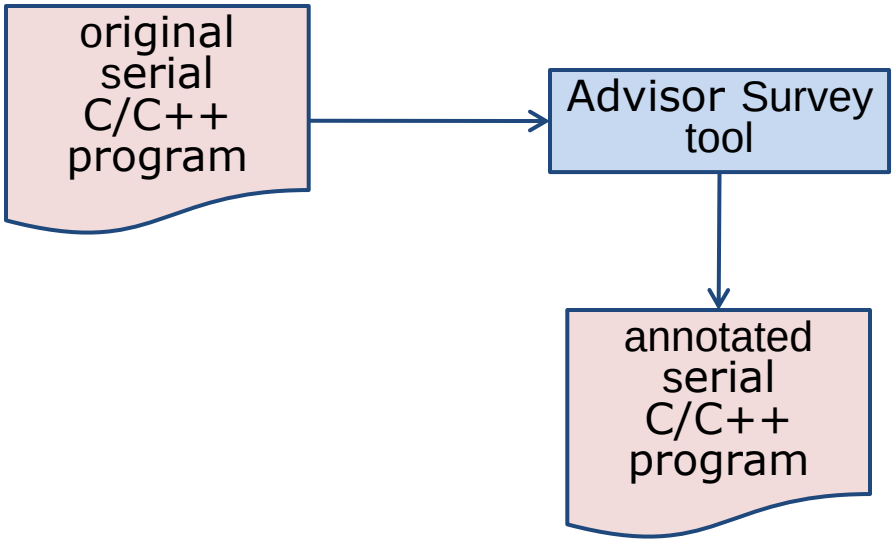


Overview – Intel Parallel Advisor



Run Survey tool to measure where your application is spending time

Overview – Intel Parallel Advisor

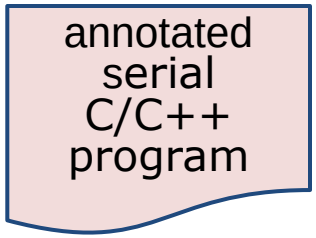


Run Survey tool to measure where your application is spending time

Add annotations to describe how you might divide the work of the hot-spot into parallel tasks

Overview – Intel Parallel Advisor

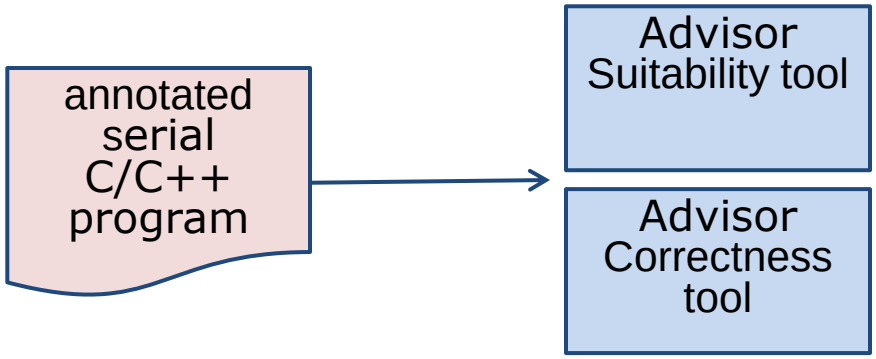
The annotated program still has serial semantics. Working on it is just normal programming.



Overview – Intel Parallel Advisor

The annotated program still has serial semantics. Working on it is just normal programming.

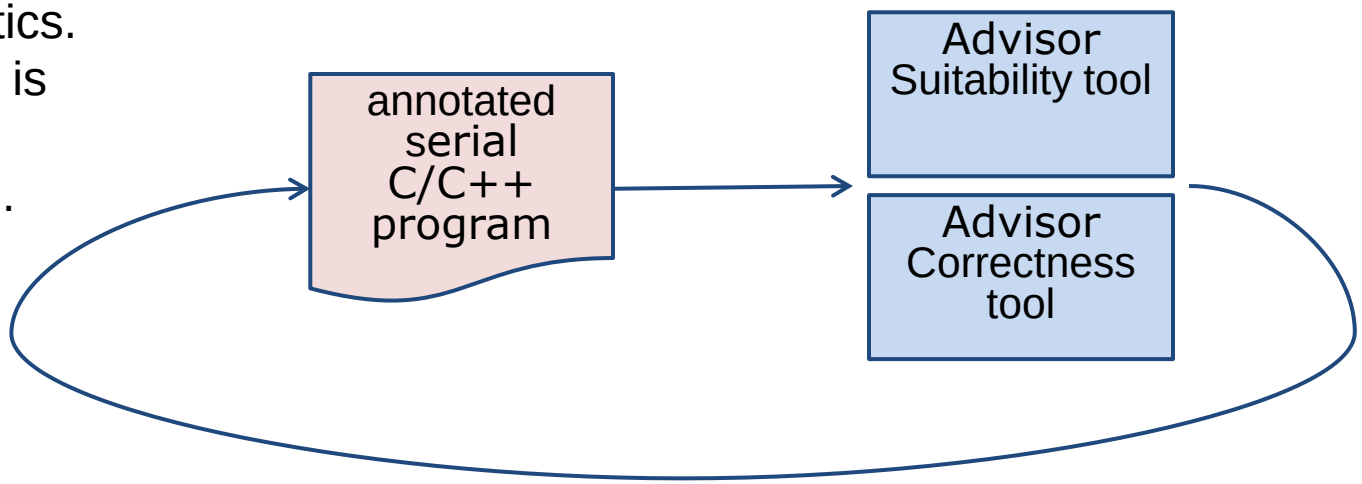
Use the performance modeling (suitability) and race detection (Correctness) tools to identify potential issues



Overview – Intel Parallel Advisor

Use the performance modeling (suitability) and race detection (Correctness) tools to identify potential issues

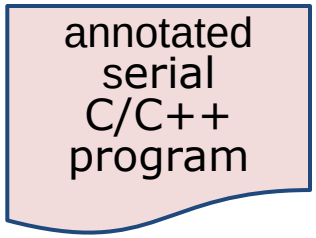
The annotated program still has serial semantics. Working on it is just normal programming.



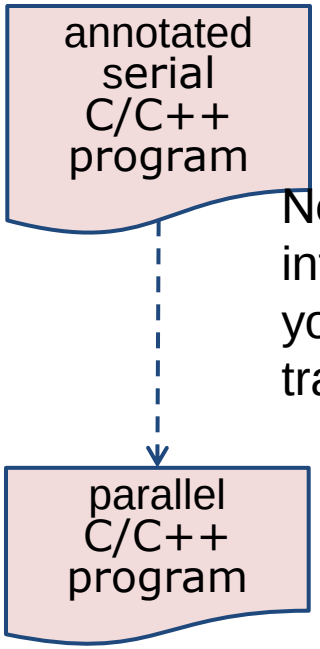
Adjust the annotations or the program code itself to resolve performance and correctness issues

Overview – Intel Parallel Advisor

The final version of the annotated program is projected to have good performance and no races.



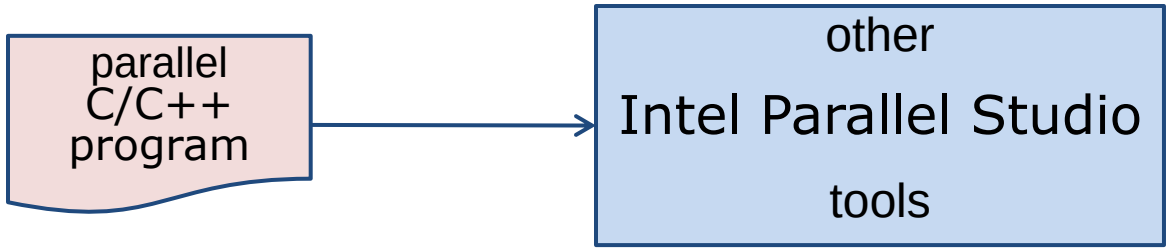
Overview – Intel Parallel Advisor



Now, it's time to change the annotations into real parallel code – TBB or Cilk or your favorite model. This straightforward translation is done by the programmer.

Overview – Intel Parallel Advisor

Use the other tools in Intel Parallel Studio to work with your parallel application (compile, debug, tune, etc)



Agenda

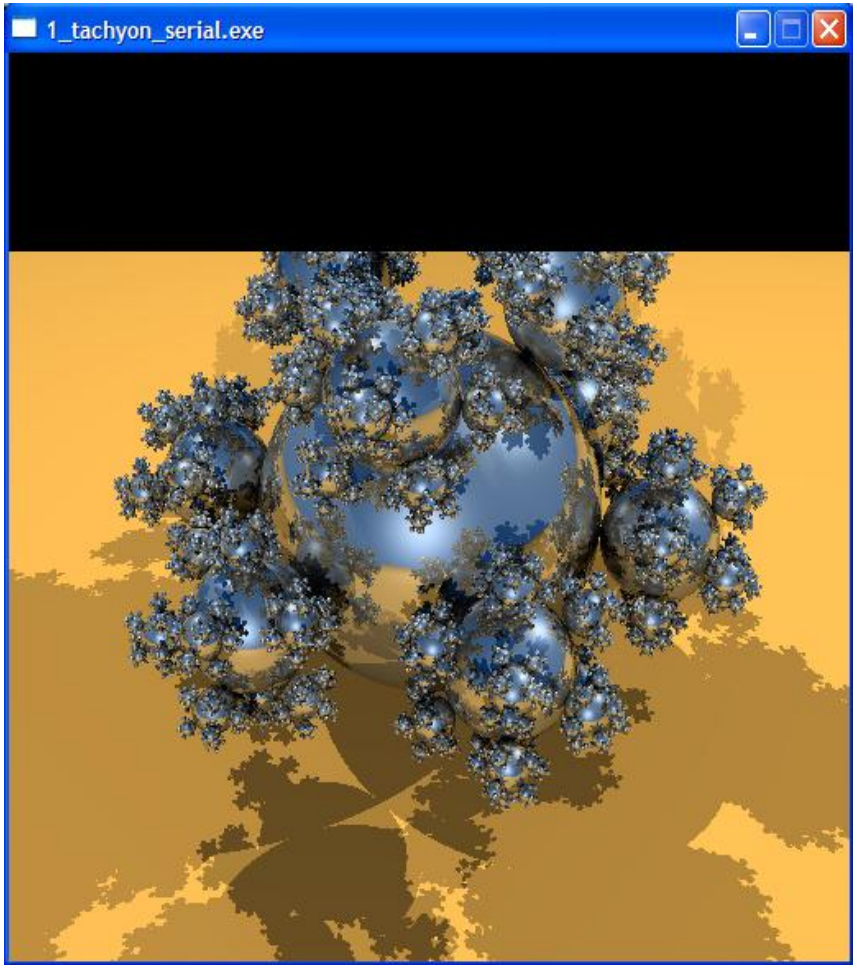
- Overview
- Find where parallelism can be usefully added to a program
- Fix the parts of the program that prevent parallelism
- Add parallelism
- Test the revised program for correctness and performance
- Don’t panic

Finding where to add parallelism

- Find steps which are using lots of time
- Measure the number of steps needed to make a good size task
- Find how the steps interact

Analyze, don't guess.
Computers are good at analysis.

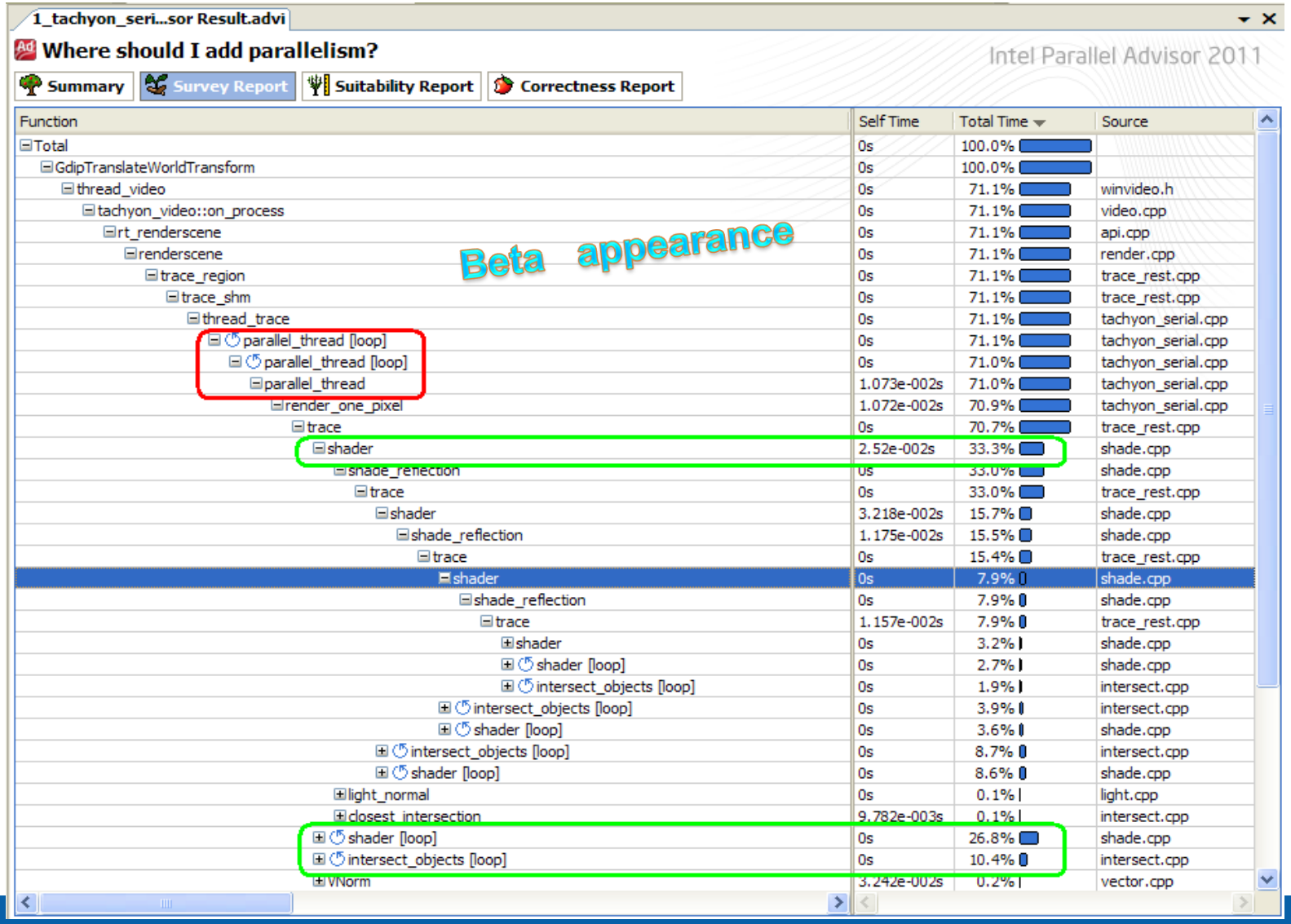
Tachyon Ray-tracer: Serial



Finding where to add parallelism

- Find steps which are using lots of time
  - Profile the serial algorithm : production build+ data
  - Find outer calls/loops that take lots of time, and that the architect thinks could be done in parallel

Survey Report – Profile: Hot Call Chain(s)



Function	Self Time	Total Time	Source
Total	0s	100.0%	
GdipTranslateWorldTransform	0s	100.0%	
thread_video	0s	71.1%	winvideo.h
tachyon_video::on_process	0s	71.1%	video.cpp
rt_renderscene	0s	71.1%	api.cpp
renderscene	0s	71.1%	render.cpp
trace_region	0s	71.1%	trace_rest.cpp
trace_shm	0s	71.1%	trace_rest.cpp
thread_trace	0s	71.1%	tachyon_serial.cpp
parallel_thread [loop]	0s	71.1%	tachyon_serial.cpp
parallel_thread [loop]	0s	71.0%	tachyon_serial.cpp
parallel_thread	1.073e-002s	71.0%	tachyon_serial.cpp
render_one_pixel	1.072e-002s	70.9%	tachyon_serial.cpp
trace	0s	70.7%	trace_rest.cpp
shader	2.52e-002s	33.3%	shade.cpp
shade_reflection	0s	33.0%	shade.cpp
trace	0s	33.0%	trace_rest.cpp
shader	3.218e-002s	15.7%	shade.cpp
shade_reflection	1.175e-002s	15.5%	shade.cpp
trace	0s	15.4%	trace_rest.cpp
shader	0s	7.9%	shade.cpp
shade_reflection	0s	7.9%	shade.cpp
trace	1.157e-002s	7.9%	trace_rest.cpp
shader	0s	3.2%	shade.cpp
shader [loop]	0s	2.7%	shade.cpp
intersect_objects [loop]	0s	1.9%	intersect.cpp
intersect_objects [loop]	0s	3.9%	intersect.cpp
shader [loop]	0s	3.6%	shade.cpp
intersect_objects [loop]	0s	8.7%	intersect.cpp
shader [loop]	0s	8.6%	shade.cpp
light_normal	0s	0.1%	light.cpp
closest_intersection	9.782e-003s	0.1%	intersect.cpp
shader [loop]	0s	26.8%	shade.cpp
intersect_objects [loop]	0s	10.4%	intersect.cpp
VNorm	3.242e-002s	0.2%	vector.cpp

Finding where to add parallelism

- Annotate the chosen sites and tasks
  - Sites contain where the tasks start, won't exit until the contained tasks end
  - Tasks are modeled as if they run in parallel

```
for (int s2 = 0; s2 < s2Count; s2++) {
    ANNOTATE_SITE_BEGIN(s2)
        delay();
        char lock;
        for (int t2 = 0; t2 < t2Count; t2++) {
            ANNOTATE_TASK_BEGIN(t2)
```

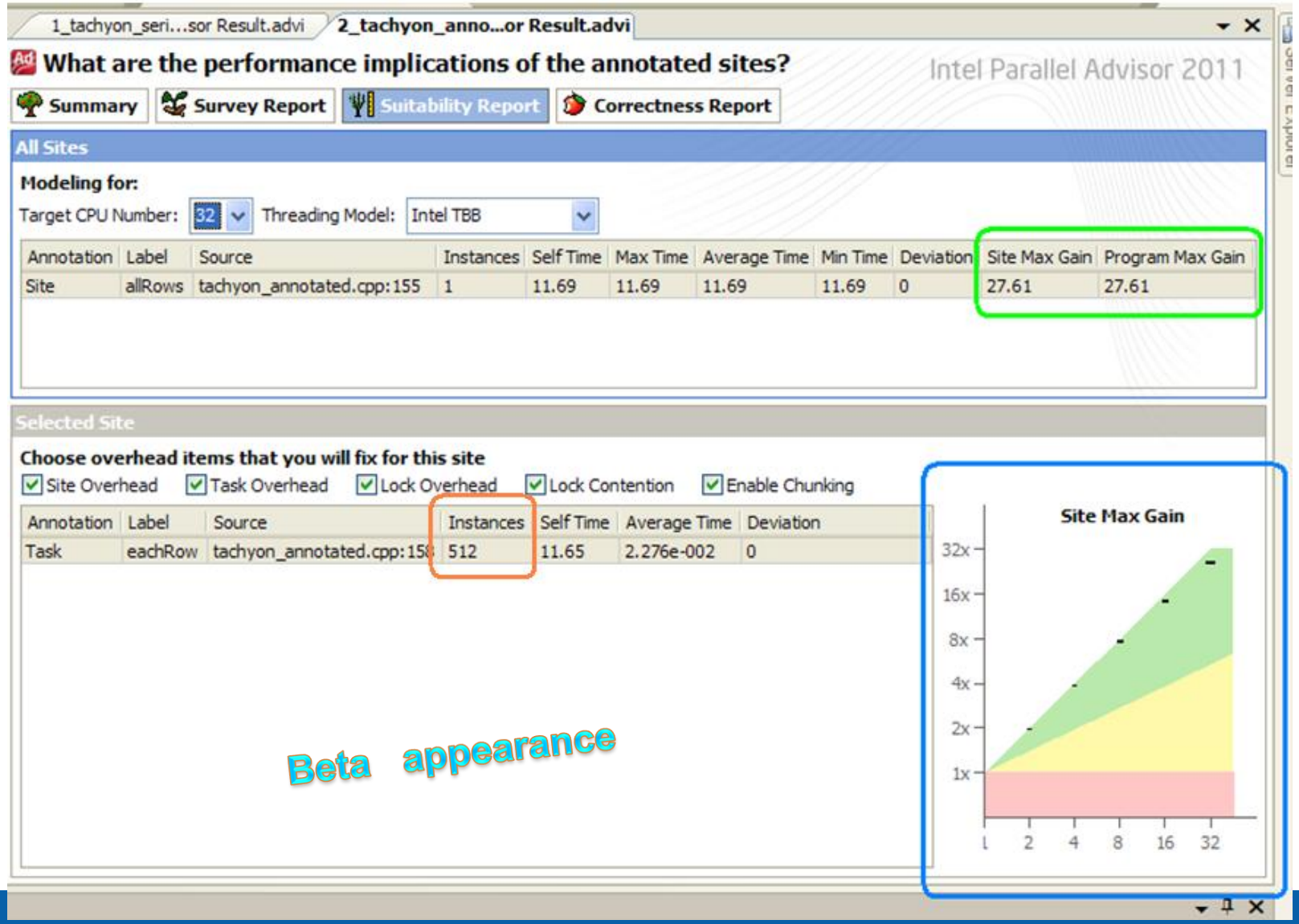
Macro use expands to a call to a stub function that can be hooked

```
                ANNOTATE_LOCK_ACQUIRE(&lock)
                delay();
                ANNOTATE_LOCK_RELEASE(&lock)
            }
            ANNOTATE_TASK_END(t2)
        }
    ANNOTATE_SITE_END(s2)
}
```

Finding where to add parallelism

- Measure the site and task times
  - Profile the annotated algorithm : release program + data
  - Annotations in bad places may slow execution
- How the measurement is done
  - Runtime binary instrumentation (Pin)
    - Intercept the calls to the stub functions
  - Measure the times between the calls
    - Uses the QPC() function and *_rdtsc* instruction
  - Compress and write to disk
    - Same as the other Intel Parallel Studio tools
  - Analyze
    - To understand the counts and times of the site, task, and lock execution

Suitability Report – *Model* Parallel Performance



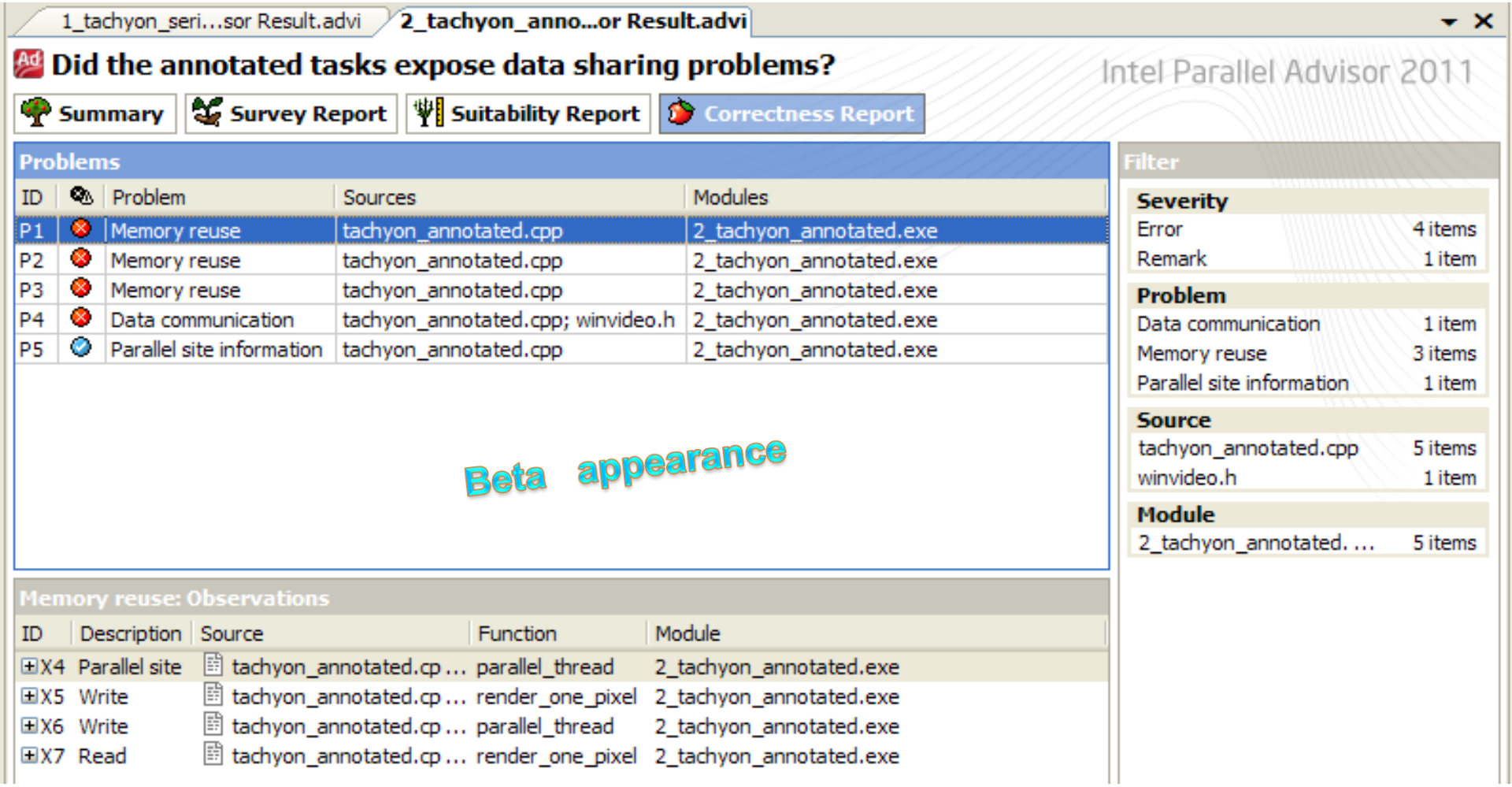
Agenda

- Overview
- Find where parallelism can be usefully added to a program
- Fix the parts of the program that prevent parallelism
- Add parallelism
- Test the revised program for correctness and performance
- Don’t panic

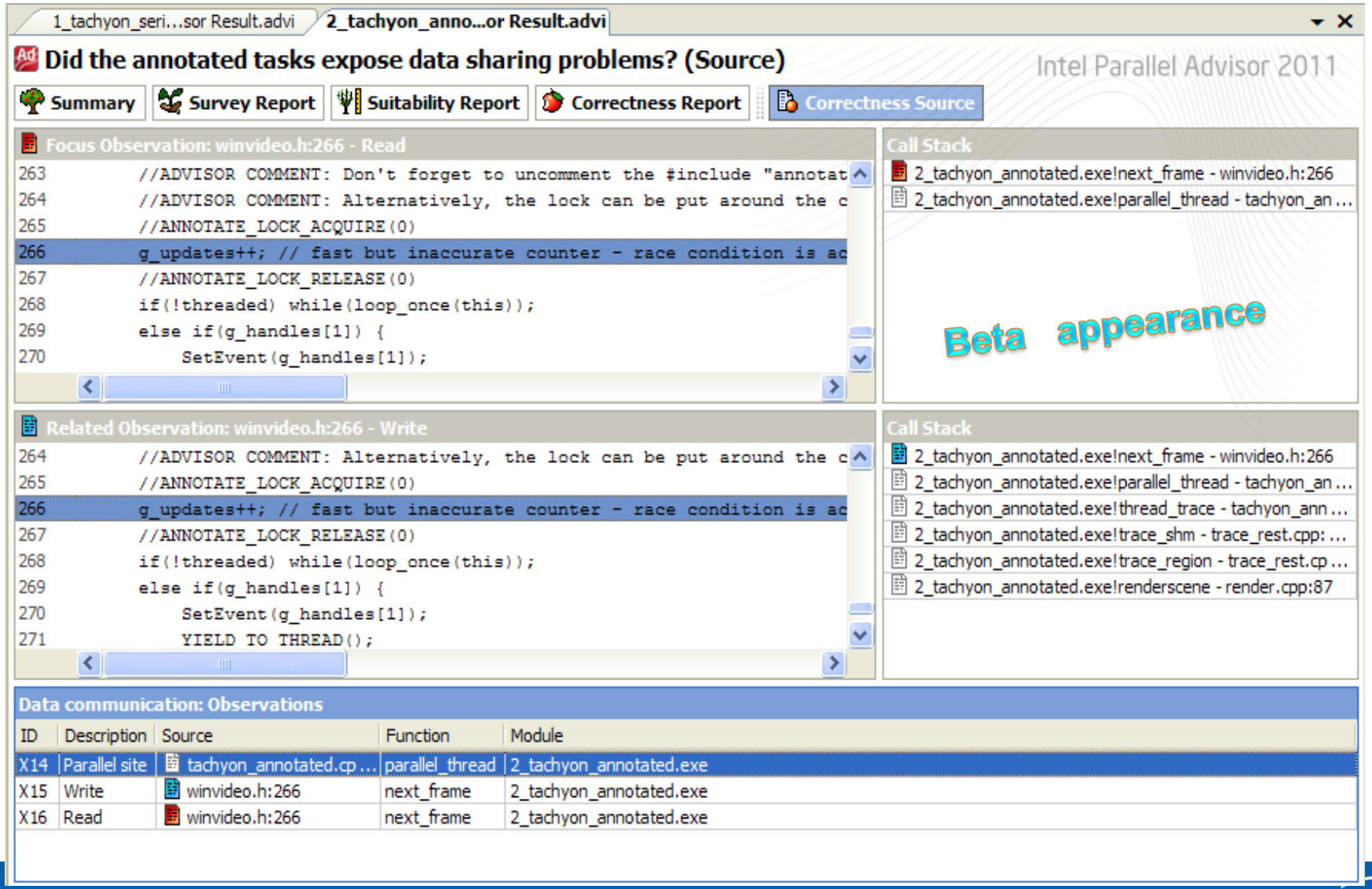
Finding code preventing parallelism

- Intel® Parallel Advisor runs the annotated *serial* program
  - It is serial, so violations of data dependencies don't crash the run
- Binary instrumentation (Pin) watches memory traffic and annotations
  - Use small data sets
    - Slow-down can be substantial
- Analyzes the data to determine what could go wrong when the tasks are done in parallel
  - Use debug builds, so problems can be mapped to sources
- Only memory sharing issues and locking problems are found

Correctness Report – *Model* Parallel Behavior



Source for *Data Communication* Error (Race)



Resolving roadblocks to parallelism

- Change the serial code to remove these
  - Debug these changes like you do any other change
- Learn about and use existing parallelized libraries; or thread-safe libraries
- Use per-task resources rather than shared resources
  - local variables, own files, other resources
- Use reductions
  - Supported directly by Cilk, Intel® TBB, OpenMP™
- Use locks only if none of the above are suitable

Fix the issues in the serial code

Agenda

- Overview
- Find where parallelism can be usefully added to a program
- Fix the parts of the program that prevent parallelism
- Add parallelism
- Test the revised program for correctness and performance
- Don’t panic

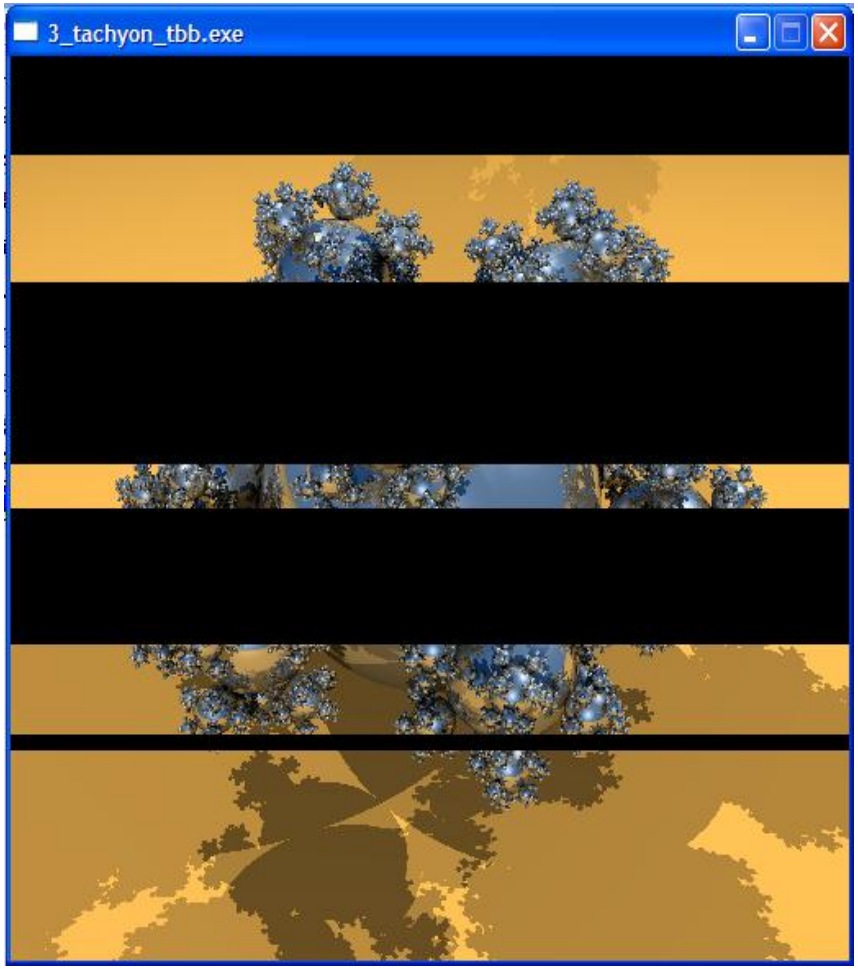
Adding parallelism

- Choose your parallel framework
  - Will depend on your programming language, portability needs, future growth needs
    - Cilk, Intel® Threading Building Blocks (Intel® TBB), OpenMP*, o/s threads
    - Intel® Parallel Composer supports these
- Modify your build and deployment environments
  - Compiler switches
  - Libraries may need to be installed on target machines
- Insert parallel framework statements for each annotation, implementing its "parallel" semantics

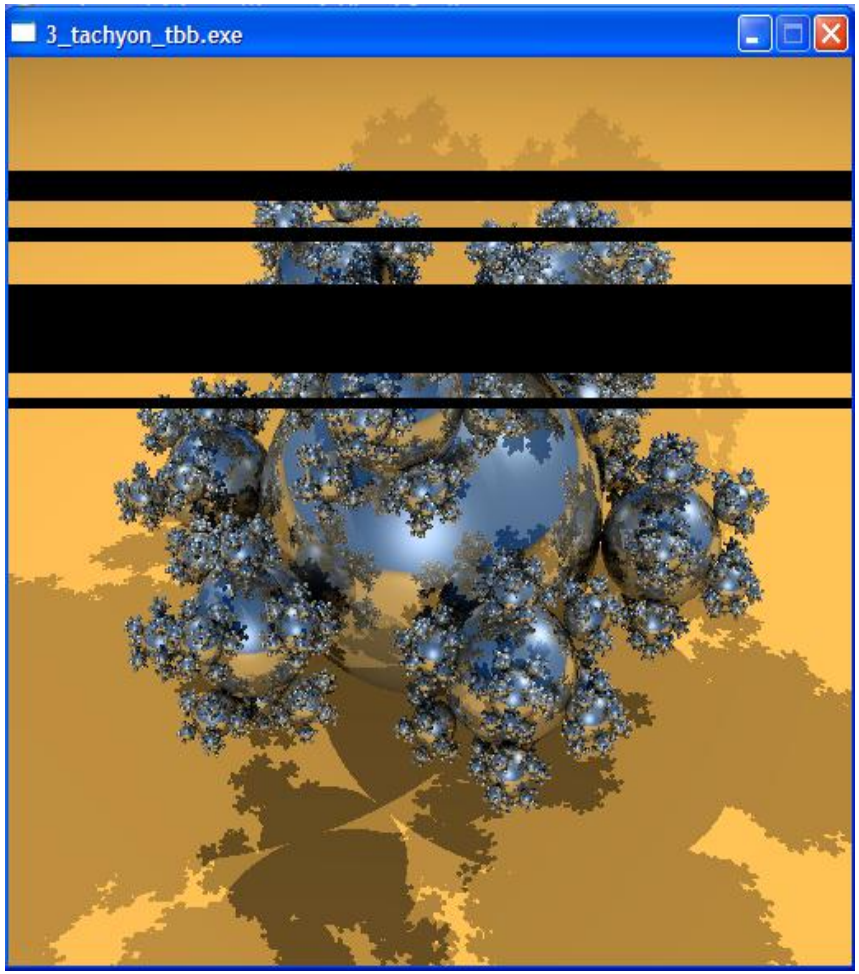
Choose the best framework for you

Tachyon – Parallel (Intel TBB), 4 cores

Appears divided into quarters



Thread stealing to finish



Agenda

- Overview
- Find where parallelism can be usefully added to a program
- Fix the parts of the program that prevent parallelism
- Add parallelism
- Test the revised program for correctness and performance
- Don’t panic

Correctness and Performance

- Parallel programs are sensitive to timing issues
  - A failure may only happen only a few times in a million runs of the program, or only on some hardware
  - Intel® Parallel Advisor correctness analysis depends on the annotations matching what you implemented
  - Intel® Parallel Inspector detects problems that may happen, even if they did not happen this time
- Parallel program timings are sensitive to task sizes, thread interactions, memory caches
  - Intel® Parallel Amplifier measures where the time is spent

Measure, don't guess

Agenda

- Overview
- Find where parallelism can be usefully added to a program
- Fix the parts of the program that prevent parallelism
- Add parallelism
- Test the revised program for correctness and performance
- Don’t panic

Don't Panic

- Parallel programming has been around for a long time, and is perceived as very difficult
- But any programmer can do it and get worthwhile benefits IF
  - They go step by step
  - They use modern frameworks and tools
  - They avoid the more complex techniques
- Intel® Parallel Advisor documentation introduces the programmer to this world

Intel® Parallel Studio helps with each step

Summary

- Analysis, preparation and testing of the serial code simplifies adding parallelism
- Adding and using parallelism requires enhanced tools

- Analyze, don't guess, where to add
- Fix the issues in the serial code
- Choose the best parallel framework for you
- Measure, don't guess, performance issues
- Intel® Parallel Studio helps with each step

Additional sources of information on this topic:

- More web based info: http://software.intel.com and http://software.intel.com/en-us/intel-learning-lab/

- Search the internet for
  - "Intel Parallel Studio"
  - "Intel Parallel Advisor"
  - "TBB"
  - "Cilk"
  - …

  - Especially anything from James Reinders

Q&A

Legal Disclaimer

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S 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 INTELLECTUAL PROPERTY RIGHT. INTEL PRODUCTS ARE NOT INTENDED FOR USE IN MEDICAL, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS.
- Intel may make changes to specifications and product descriptions at any time, without notice.
- All products, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.
- Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.
- Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.
- Intel and the Intel logo are trademarks of Intel Corporation in the United States and other countries.
- *Other names and brands may be claimed as the property of others.
- Copyright © 2010 Intel Corporation.

Risk Factors

The above statements and any others in this document that refer to plans and expectations for the first quarter, the year and the future are forward-looking statements that involve a number of risks and uncertainties. Many factors could affect Intel's actual results, and variances from Intel's current expectations regarding such factors could cause actual results to differ materially from those expressed in these forward-looking statements. Intel presently considers the following to be the important factors that could cause actual results to differ materially from the corporation's expectations. Demand could be different from Intel's expectations due to factors including changes in business and economic conditions; customer acceptance of Intel's and competitors' products; changes in customer order patterns including order cancellations; and changes in the level of inventory at customers. Intel operates in intensely competitive industries that are characterized by a high percentage of costs that are fixed or difficult to reduce in the short term and product demand that is highly variable and difficult to forecast. Additionally, Intel is in the process of transitioning to its next generation of products on 32nm process technology, and there could be execution issues associated with these changes, including product defects and errata along with lower than anticipated manufacturing yields. Revenue and the gross margin percentage are affected by the timing of new Intel product introductions and the demand for and market acceptance of Intel's products; actions taken by Intel's competitors, including product offerings and introductions, marketing programs and pricing pressures and Intel's response to such actions; defects or disruptions in the supply of materials or resources; and Intel's ability to respond quickly to technological developments and to incorporate new features into its products. The gross margin percentage could vary significantly from expectations based on changes in revenue levels; product mix and pricing; start-up costs, including costs associated with the new 32nm process technology; variations in inventory valuation, including variations related to the timing of qualifying products for sale; excess or obsolete inventory; manufacturing yields; changes in unit costs; impairments of long-lived assets, including manufacturing, assembly/test and intangible assets; the timing and execution of the manufacturing ramp and associated costs; and capacity utilization;. Expenses, particularly certain marketing and compensation expenses, as well as restructuring and asset impairment charges, vary depending on the level of demand for Intel's products and the level of revenue and profits. The majority of our non-marketable equity investment portfolio balance is concentrated in companies in the flash memory market segment, and declines in this market segment or changes in management's plans with respect to our investments in this market segment could result in significant impairment charges, impacting restructuring charges as well as gains/losses on equity investments and interest and other. Intel's results could be impacted by adverse economic, social, political and physical/infrastructure conditions in countries where Intel, its customers or its suppliers operate, including military conflict and other security risks, natural disasters, infrastructure disruptions, health concerns and fluctuations in currency exchange rates. Intel's results could be affected by the timing of closing of acquisitions and divestitures. Intel's results could be affected by adverse effects associated with product defects and errata (deviations from published specifications), and by litigation or regulatory matters involving intellectual property, stockholder, consumer, antitrust and other issues, such as the litigation and regulatory matters described in Intel's SEC reports. An unfavorable ruling could include monetary damages or an injunction prohibiting us from manufacturing or selling one or more products, precluding particular business practices, impacting our ability to design our products, or requiring other remedies such as compulsory licensing of intellectual property. A detailed discussion of these and other risk factors that could affect Intel's results is included in Intel's SEC filings, including the report on Form 10-Q.


intel®
Software