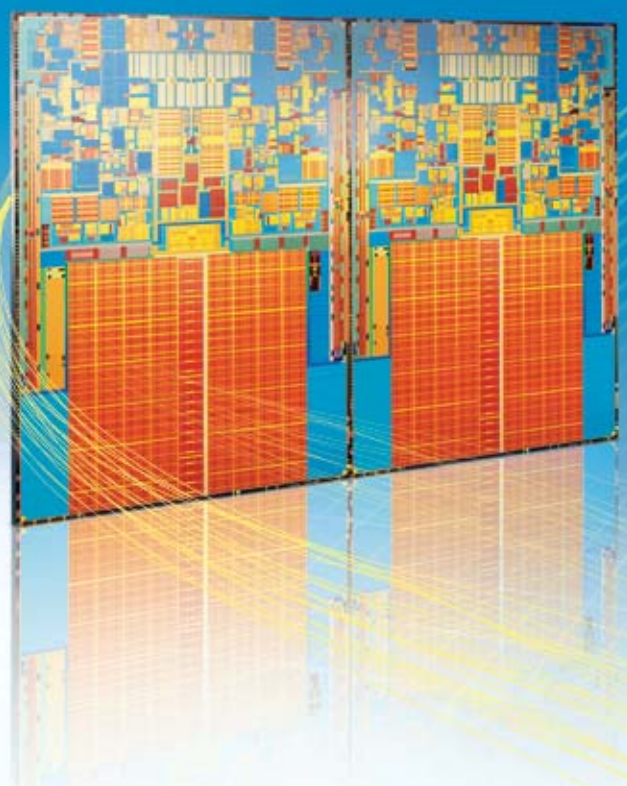
Game Threading Strategies for Next Generation Intel® Microarchitecture (Nehalem) Based Platforms



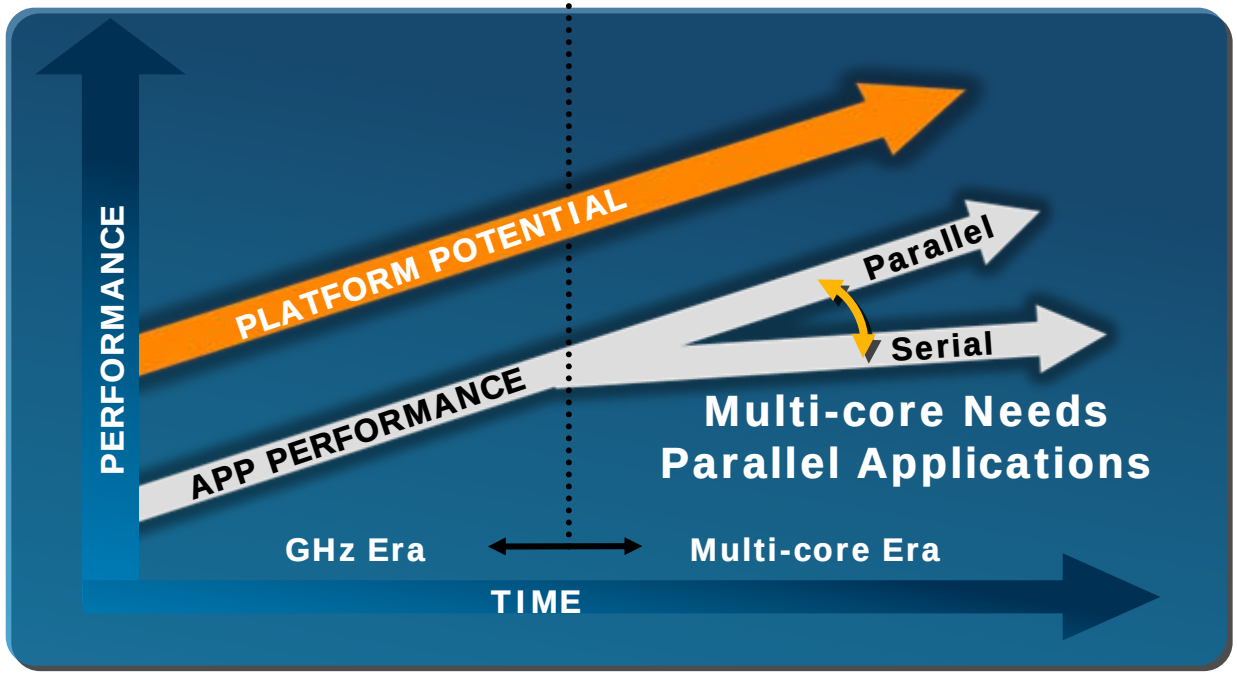
Orion Granatir
Tech Lead

VCTS002

Agenda

- Challenge
- Introducing Smoke
- Smoke Overcomes the Challenge
- Applying Smoke to Nehalem

Challenge



- The number of cores is increasing
- To really maximize performance and features, a game needs to fully utilize the CPU
- However, threading can be difficult

Threading games is hard, but worthwhile

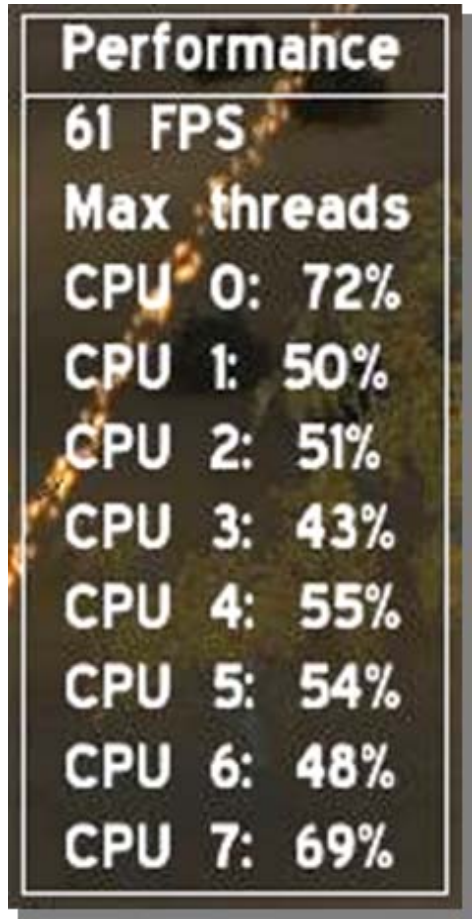
Smoke

A game framework that maxes the CPU

- Framework built by Intel for N-threads
- Demo uses real game technologies (Havok, FMOD, Ogre3D, DX9, etc.)
- Well partitioned and configurable

Why Smoke?

- Performance
  - Framework was designed to scale to N-threads
  - Discover game architectures that run well with 8 or more threads
- Prototype
  - Well partitioned and configurable
  - Easy to explore new game tech (e.g. procedural fire/smoke)
- Explore
  - Examine threading techniques
  - Understand interaction between systems
- Teach
  - Share source, demos, samples, workloads, white papers

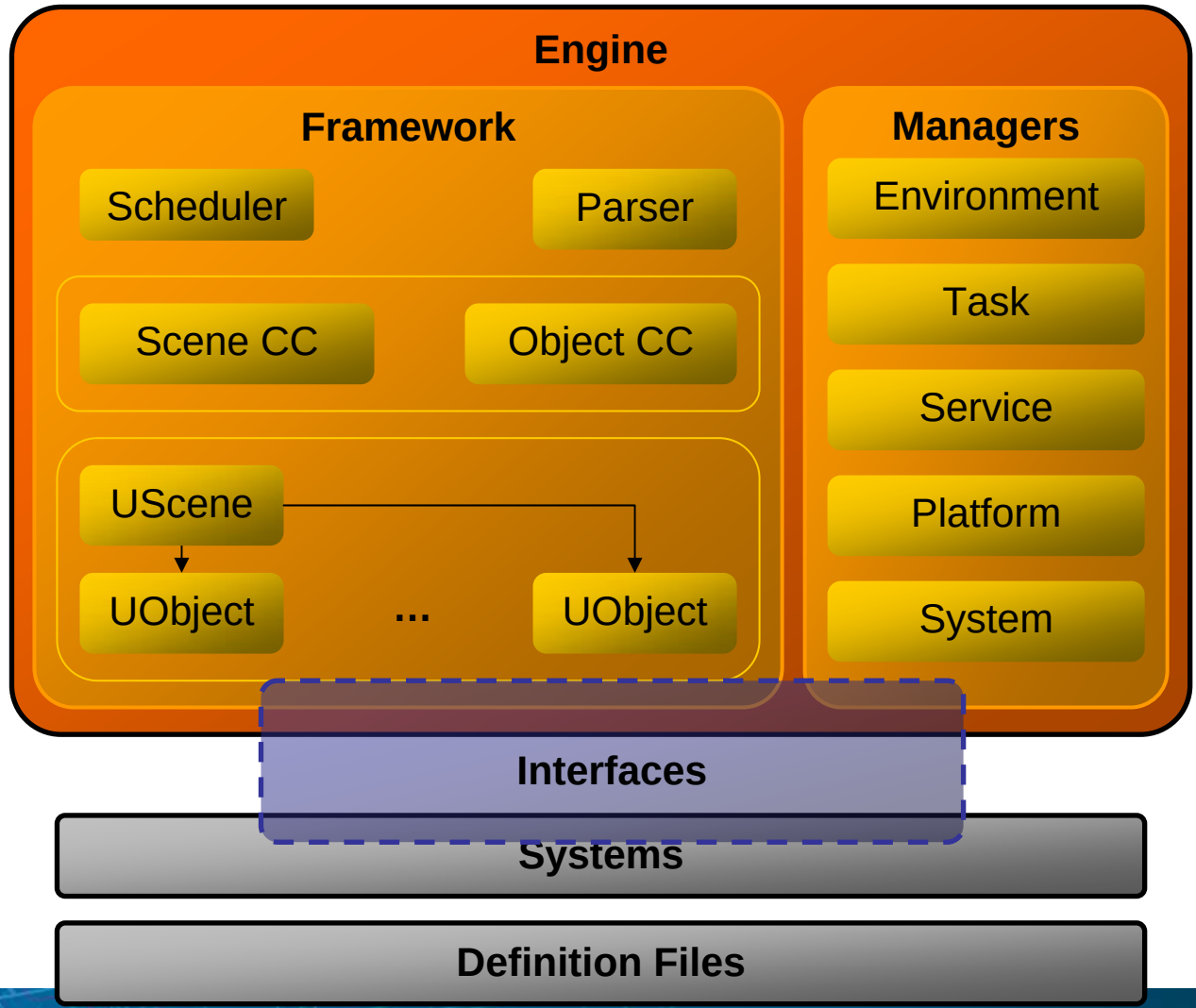


Smoke can show you how to thread effectively

The Framework

How is the Smoke highly threaded?

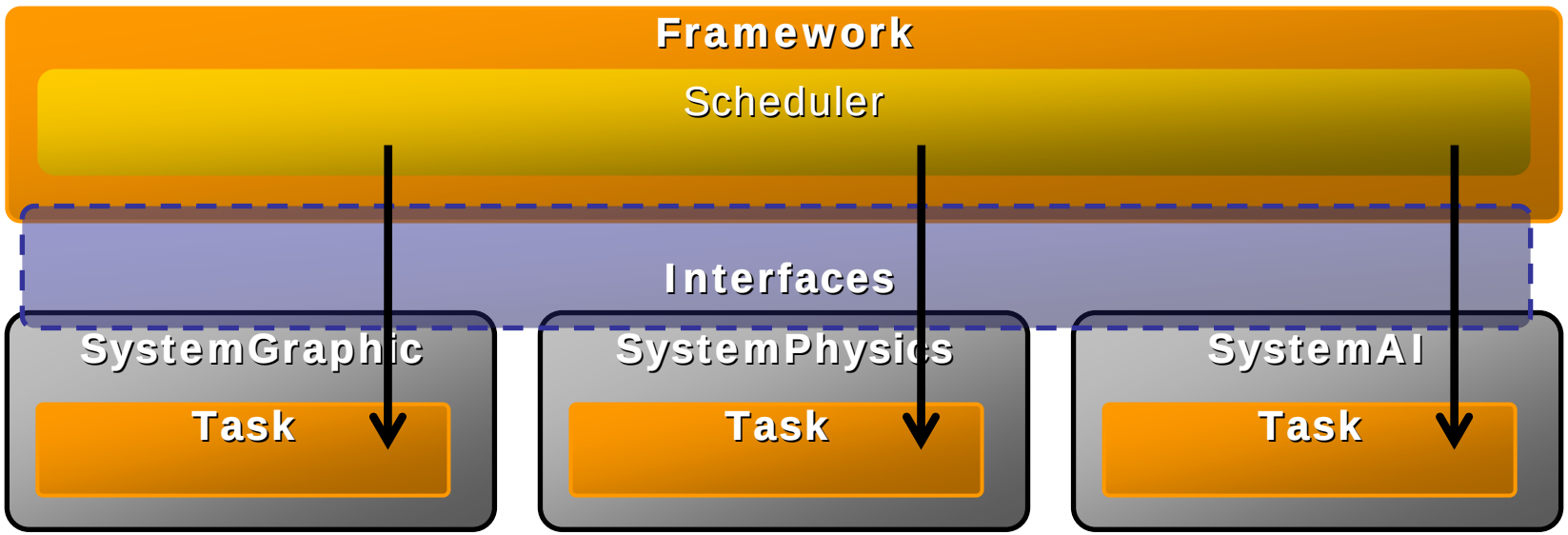
1. Scheduler manages System jobs
2. Change Control Manager minimizes thread synchronization
3. Data structured to support independent processing
4. System modularity (through interfaces)
5. Systems are specific to the demo (e.g. AI, physics, etc)



Key Design

1. Scheduler manage System Jobs

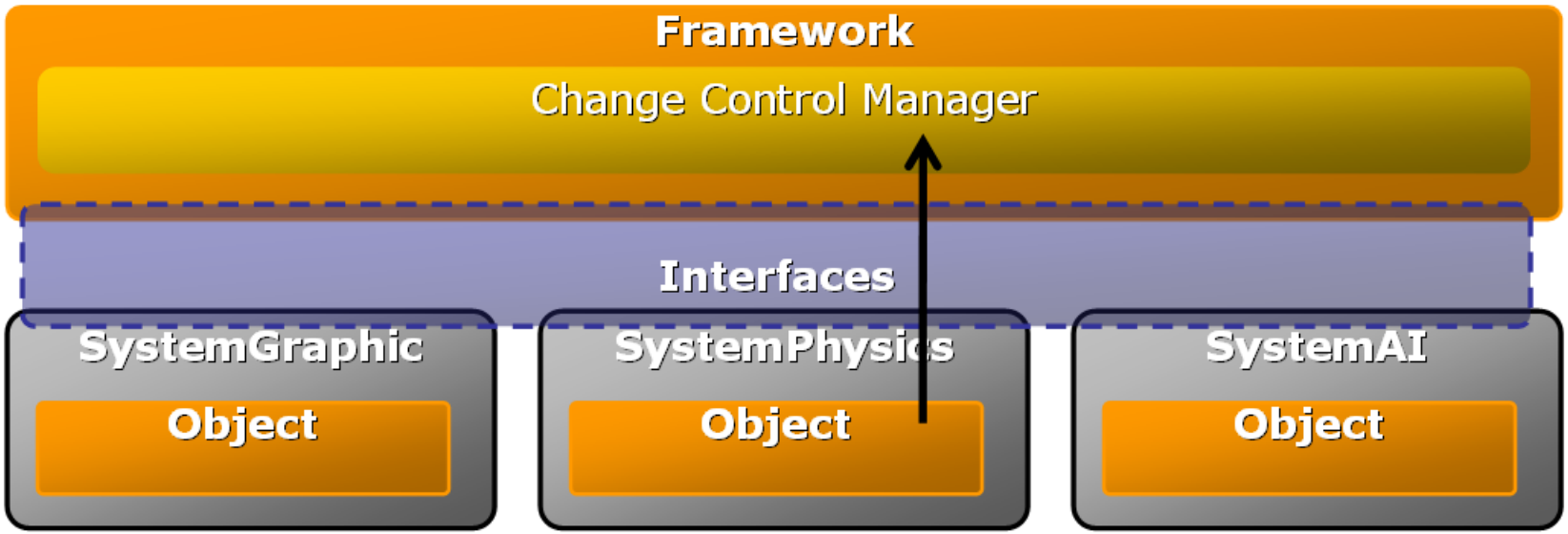
- Each System is a DLL
- Each component implements the proper interfaces to interact with the Framework (interfaces are just pure virtual base classes)
- The Scheduler will invoke the task for each system once a frame



Key Design

2. Change Control Manager minimize thread sync

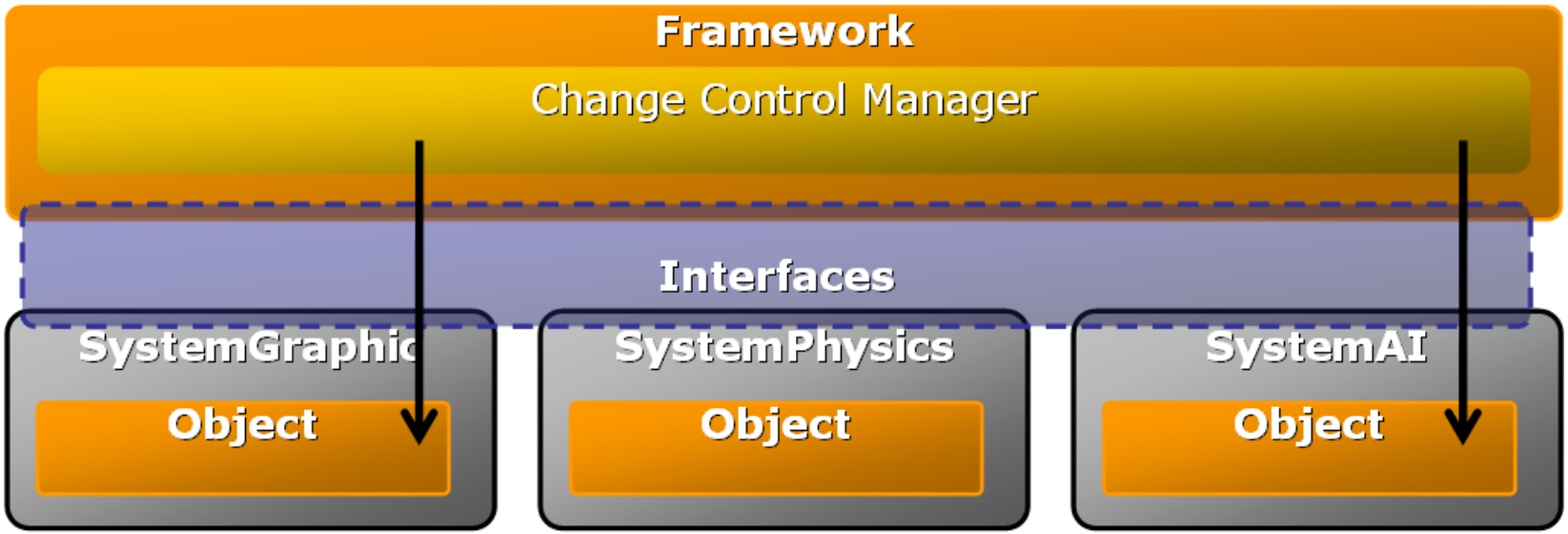
- Data is shared between the Systems with the Change Control Manager.
- If one System changes an Object within a System all Systems with a matching Object will be notified about the change.
- This allows Systems to work independently.



Key Design

2. Change Control Manager minimize thread sync

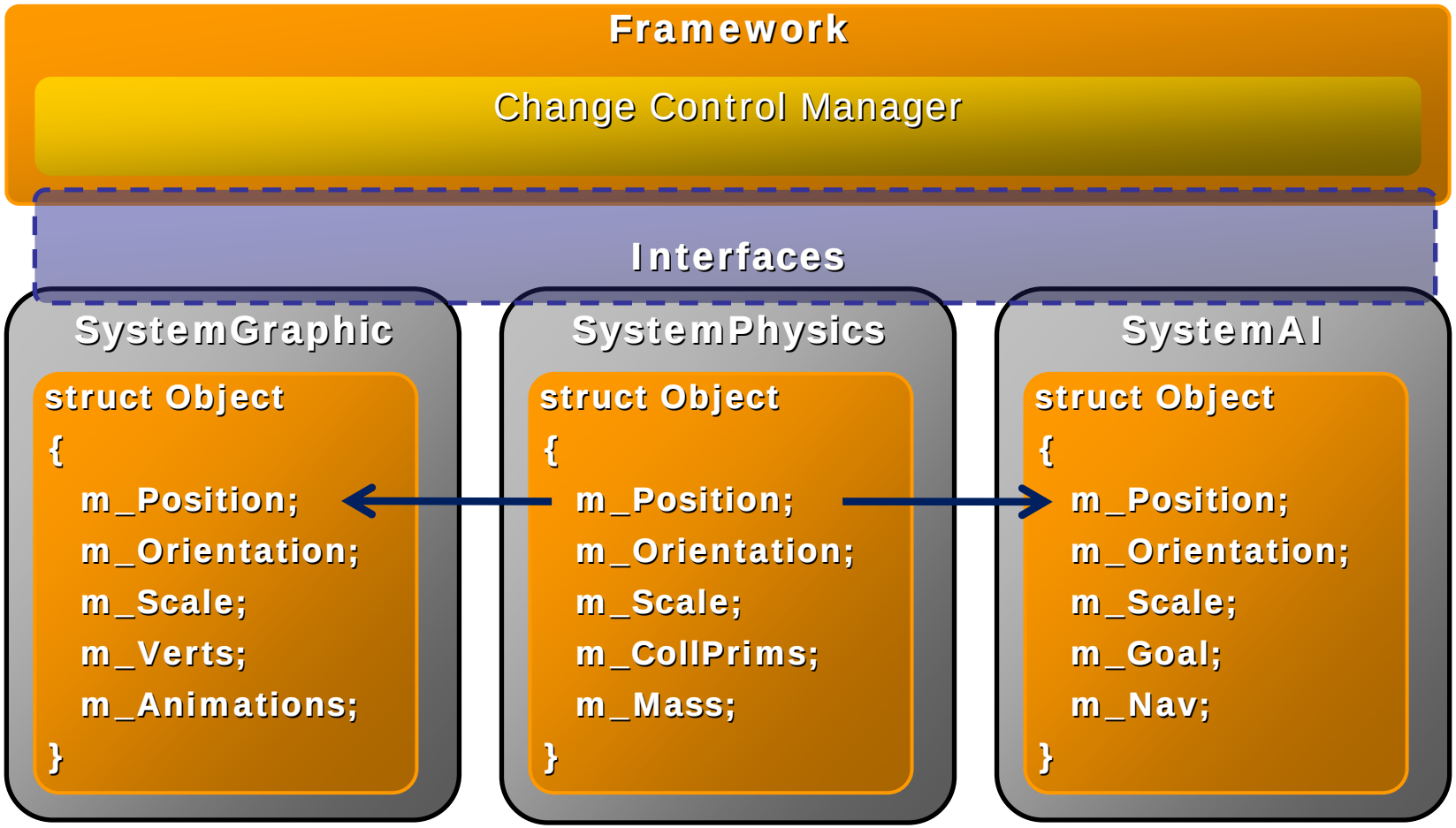
- Data is shared between the Systems with the Change Control Manager.
- If one System changes an Object within a System all Systems with a matching Object will be notified about the change.
- This allows Systems to work independently.



Key Design

3. Data structure to support independent processing

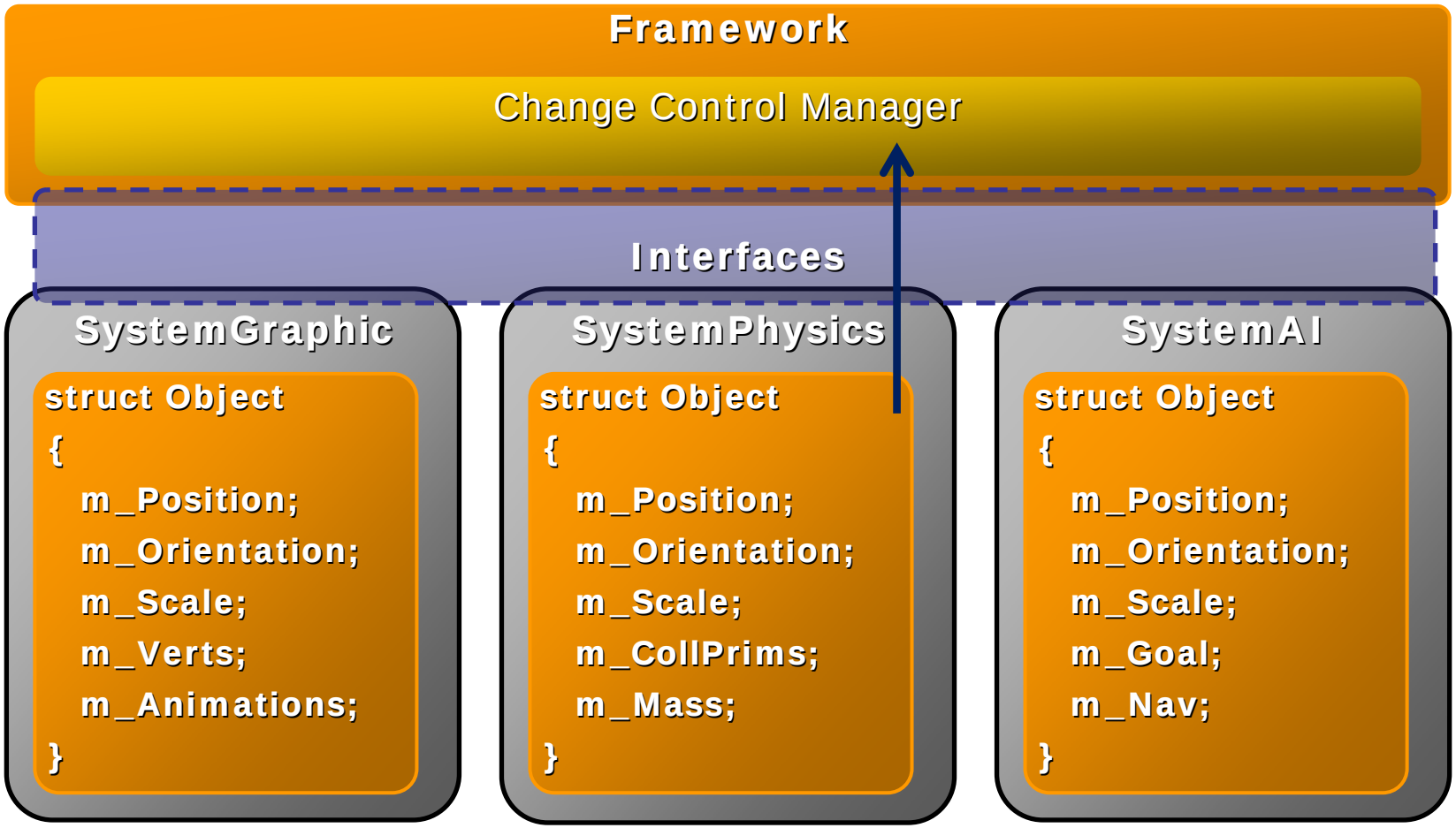
- Each System subscribes to desired changes during initialization



Key Design

Each System stores a copy of the data it needs

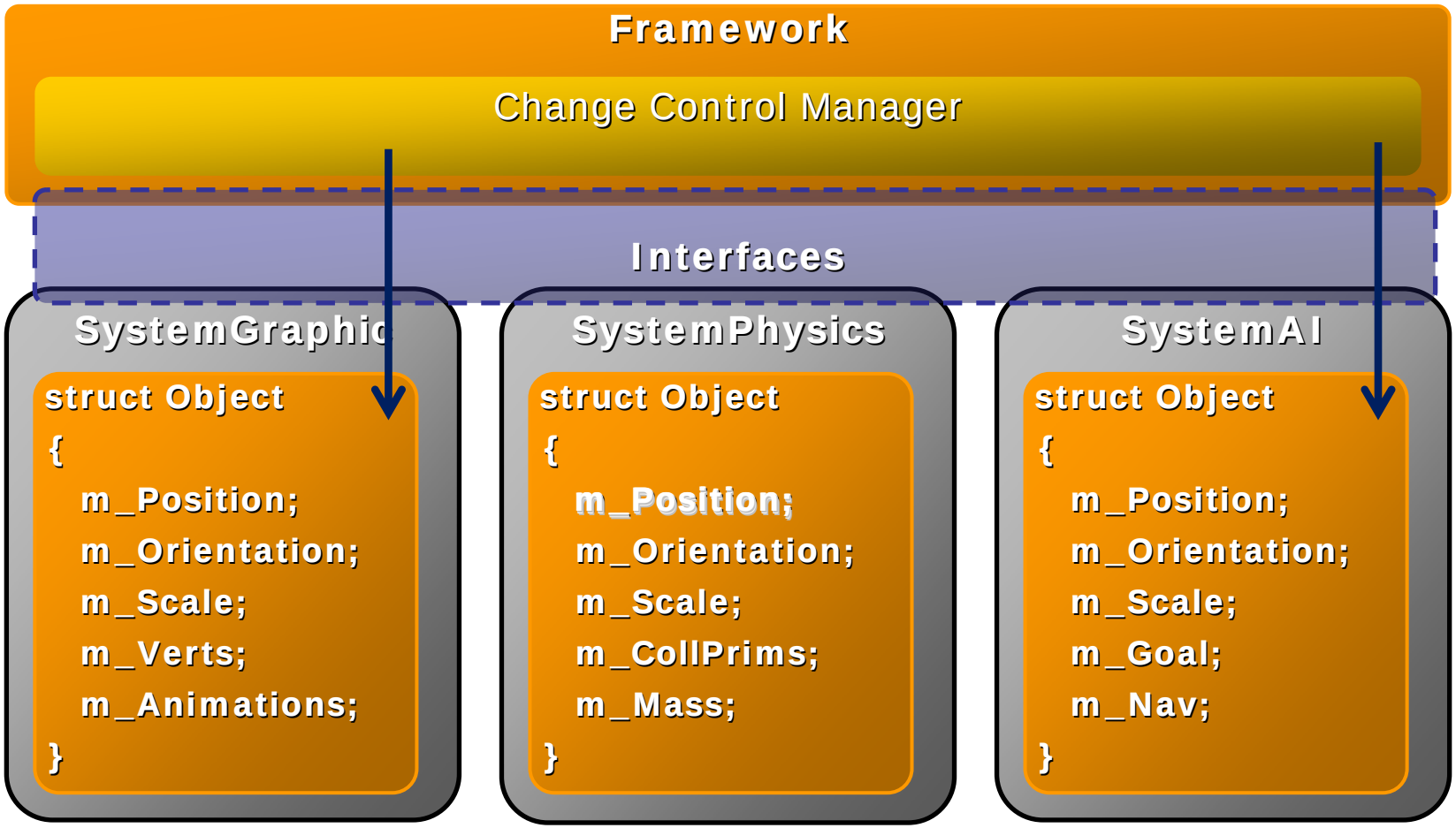
- When a System modifies its data, it tells the Change Control Manager



Key Design

Systems copy data from other Systems as needed

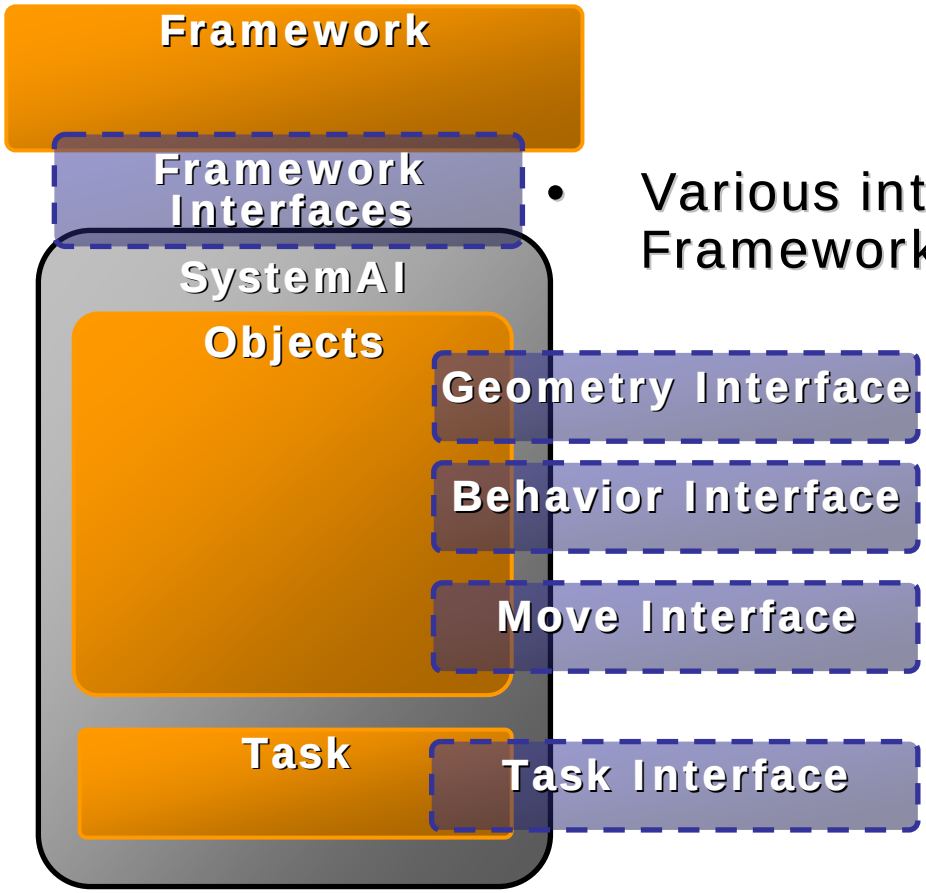
- When a System is told about a data change, it will get all the required information from the changing System



Key Design

4. System modularity (through interfaces)

- Interfaces are used to build up the interaction between Systems. This allows for high modularity.



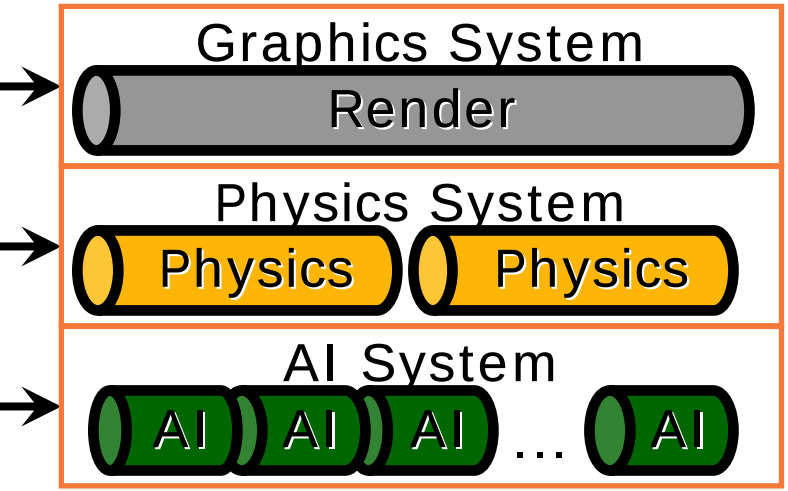
- Various interfaces to work within the Framework
- Change positions, etc
- Change behavior (generic state)
- Moving object (velocity, etc)
- Update and do work

Let's put it all together and see a frame

1. At the start of a frame, Systems run and subdivide tasks
2. Sub-tasks are added to a pool
3. Worker threads process sub-tasks
4. Tasks post changes as needed
5. Changes sent to observers at the end of the frame

A Frame

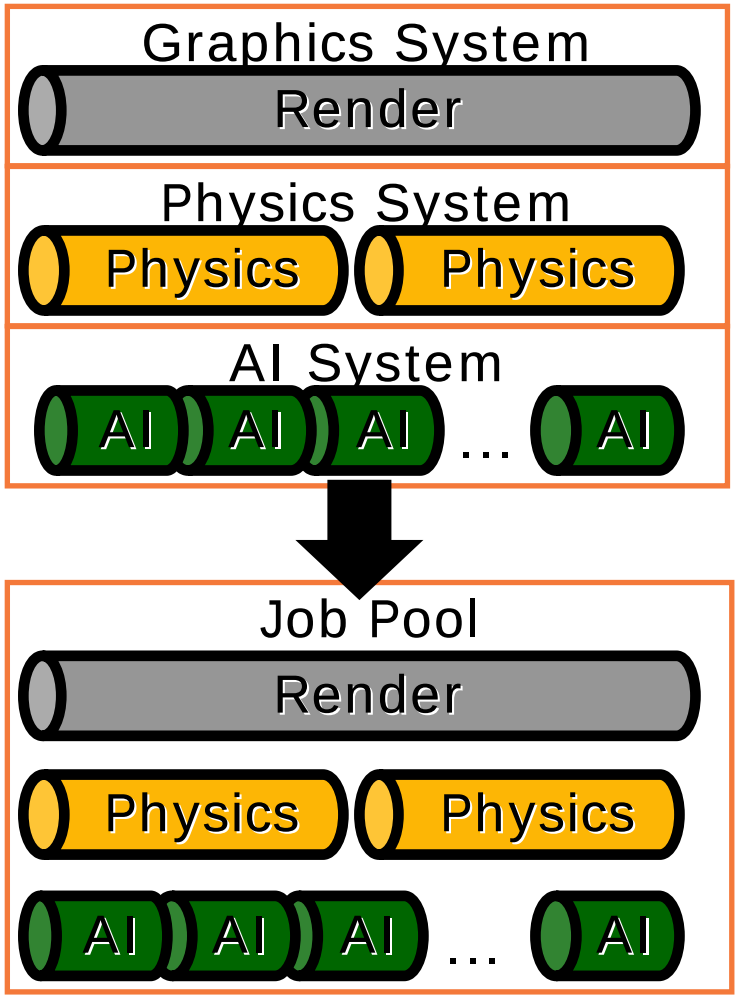
1. Systems subdivide tasks



- Scheduler invokes each system per frame
- Systems subdivide work into sub-tasks
- Systems can subdivide work based on a “natural” granularity
- Good middleware makes this easy

A Frame

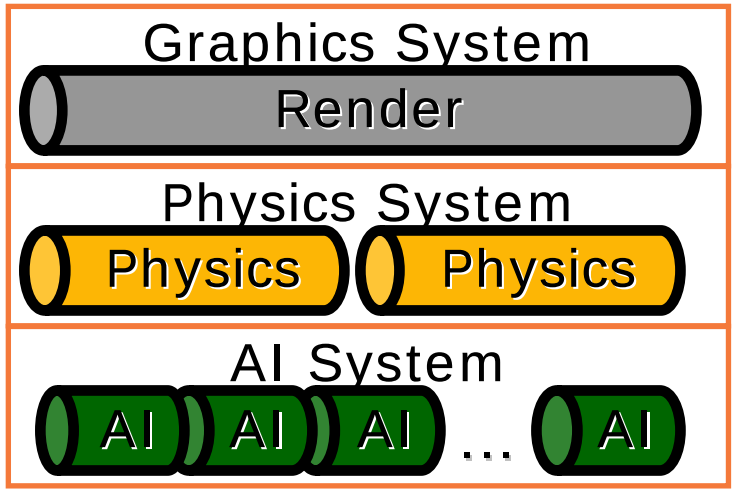
2. Add sub-tasks to a pool

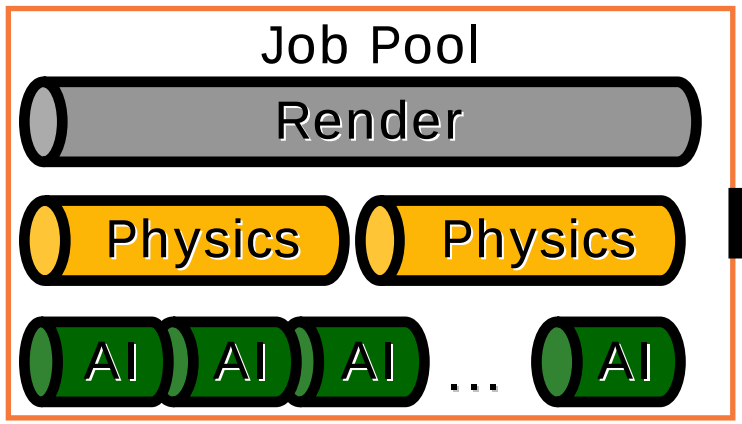


- All sub-tasks in single job pool

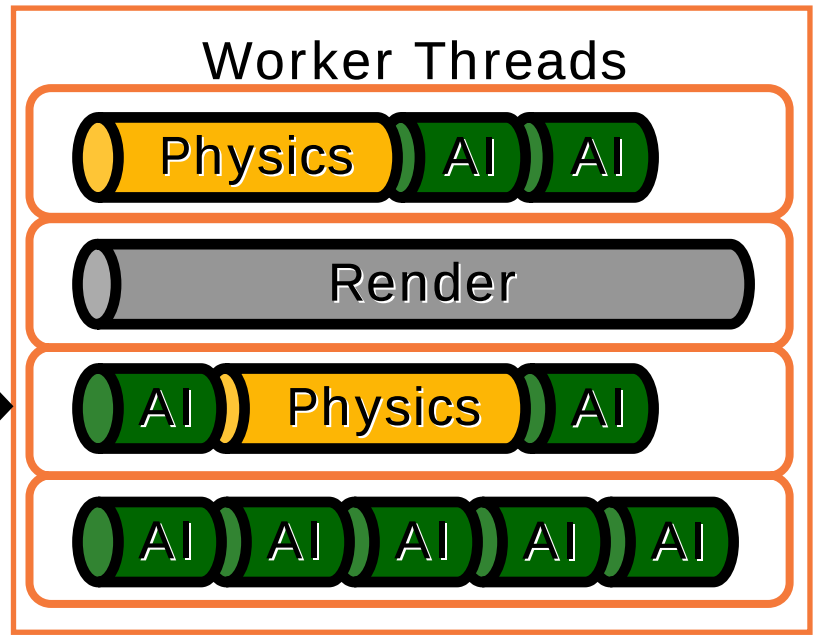
A Frame

3. Worker threads process sub-tasks



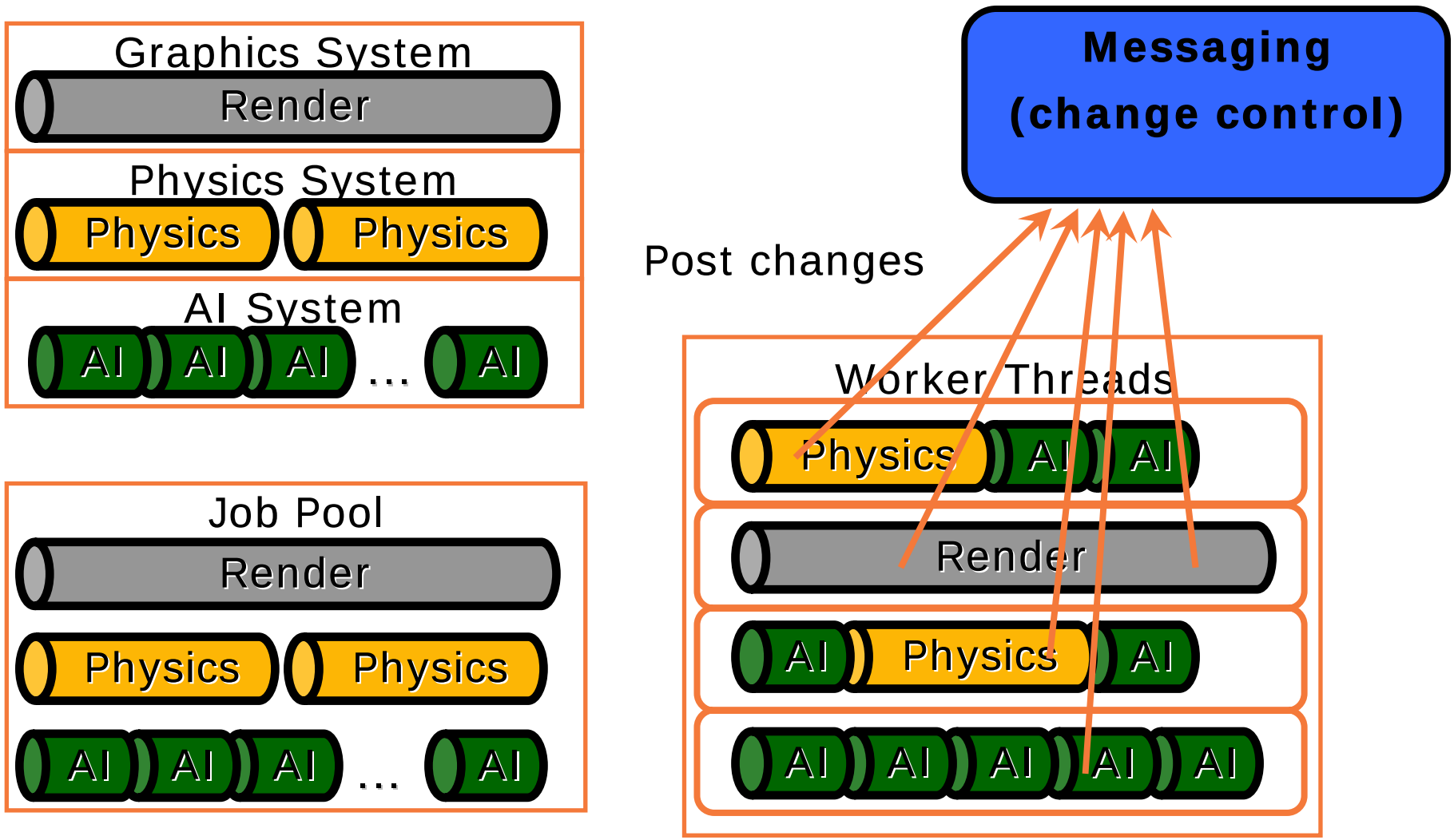


- N worker threads, 1 per core
- Sub-tasks spread out as needed



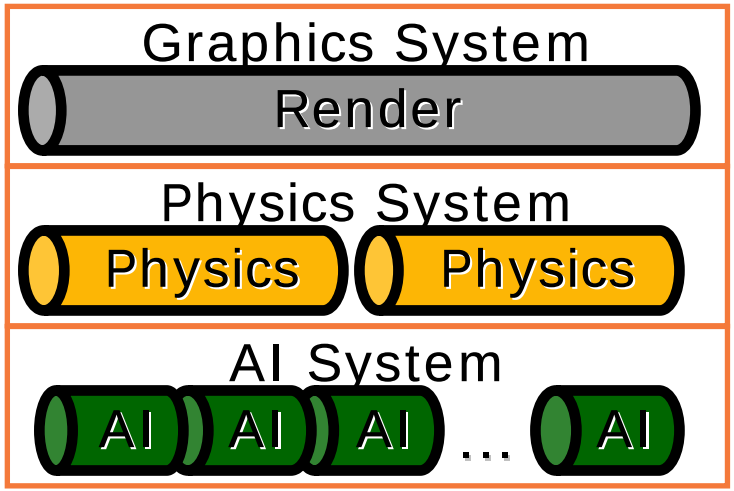
A Frame

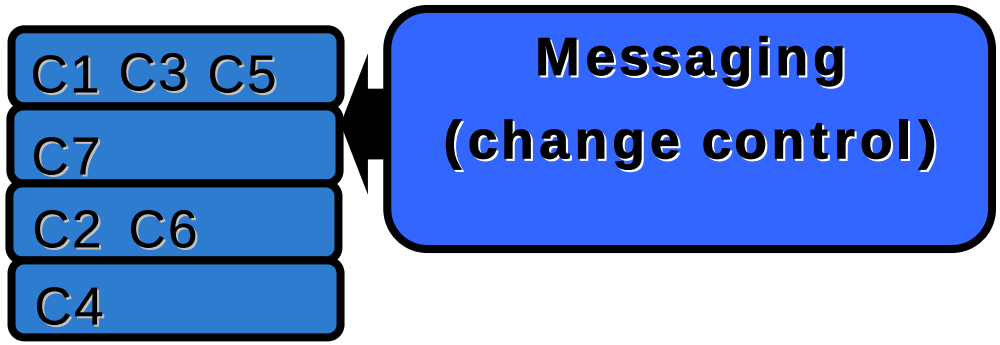
4. Tasks post changes

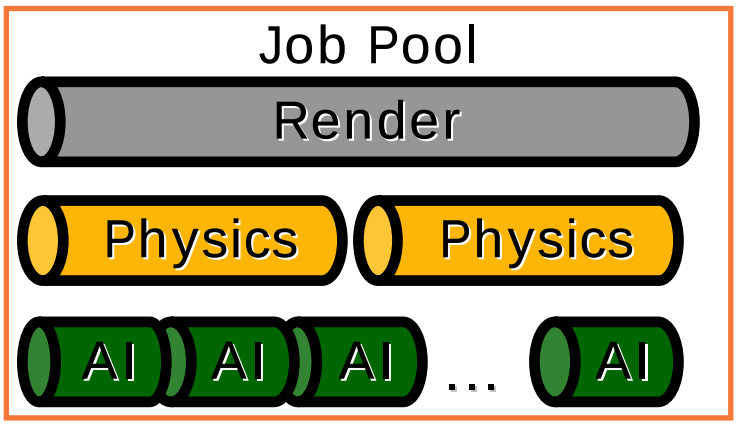


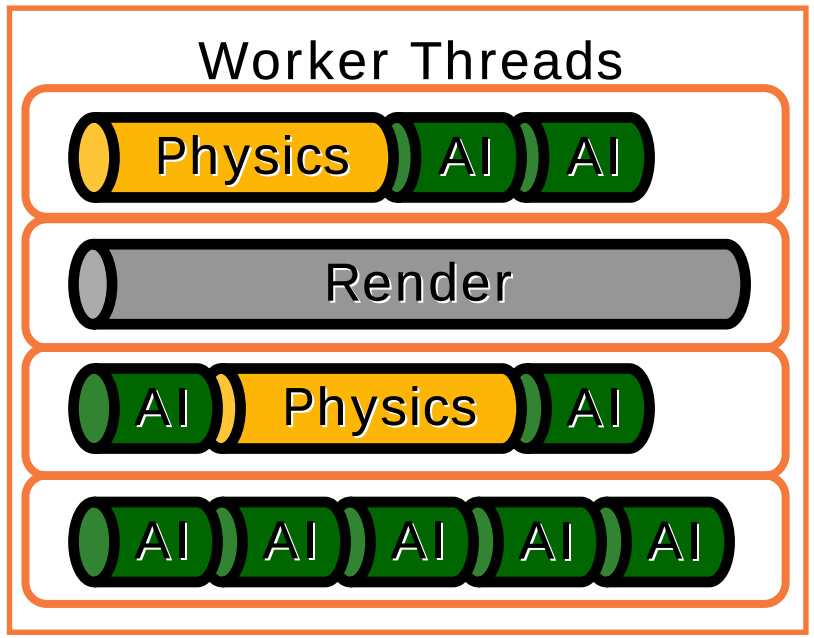
A Frame

Worker threads have unique changes queues



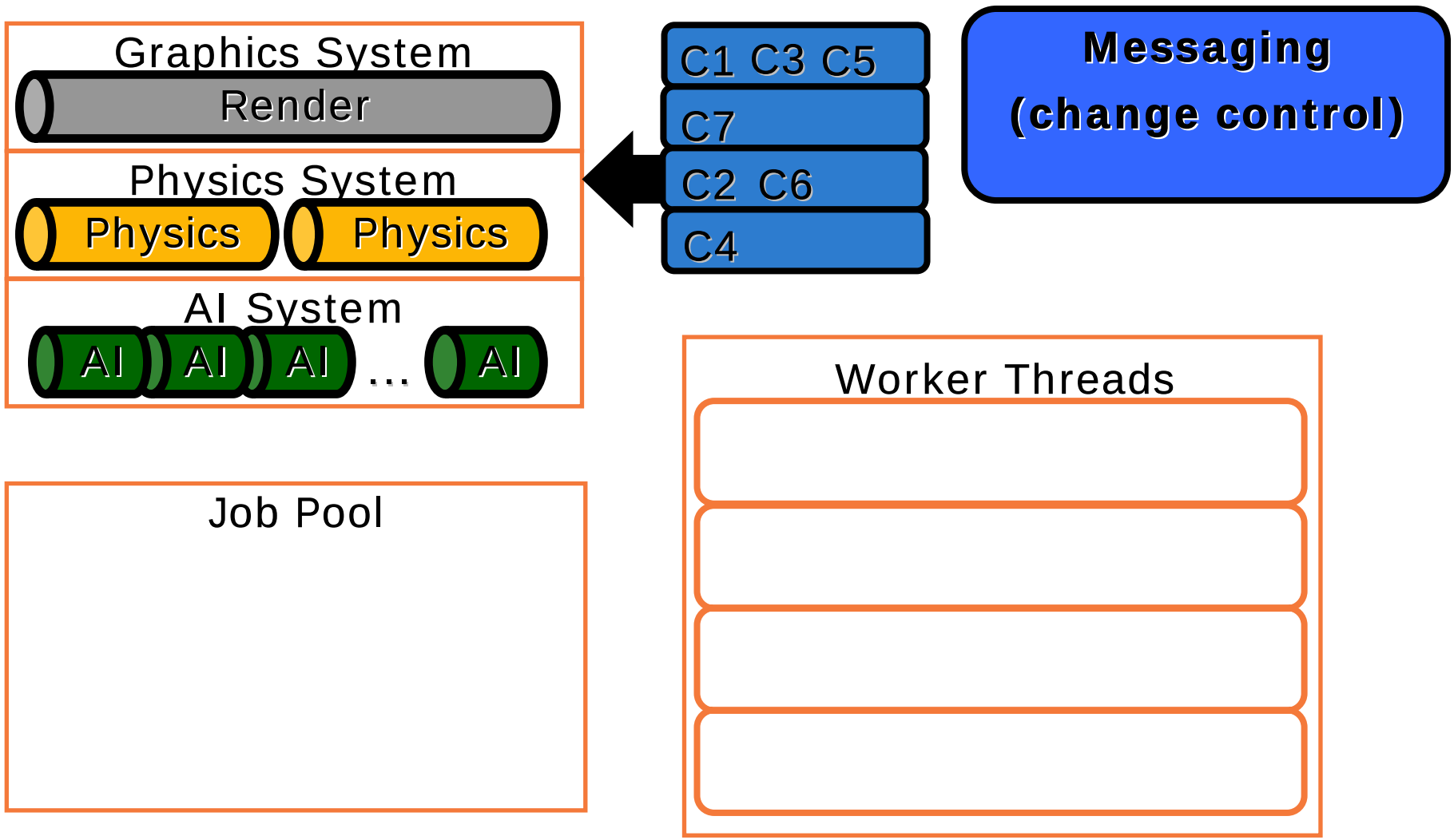






A Frame

5. Changes are sent to observers

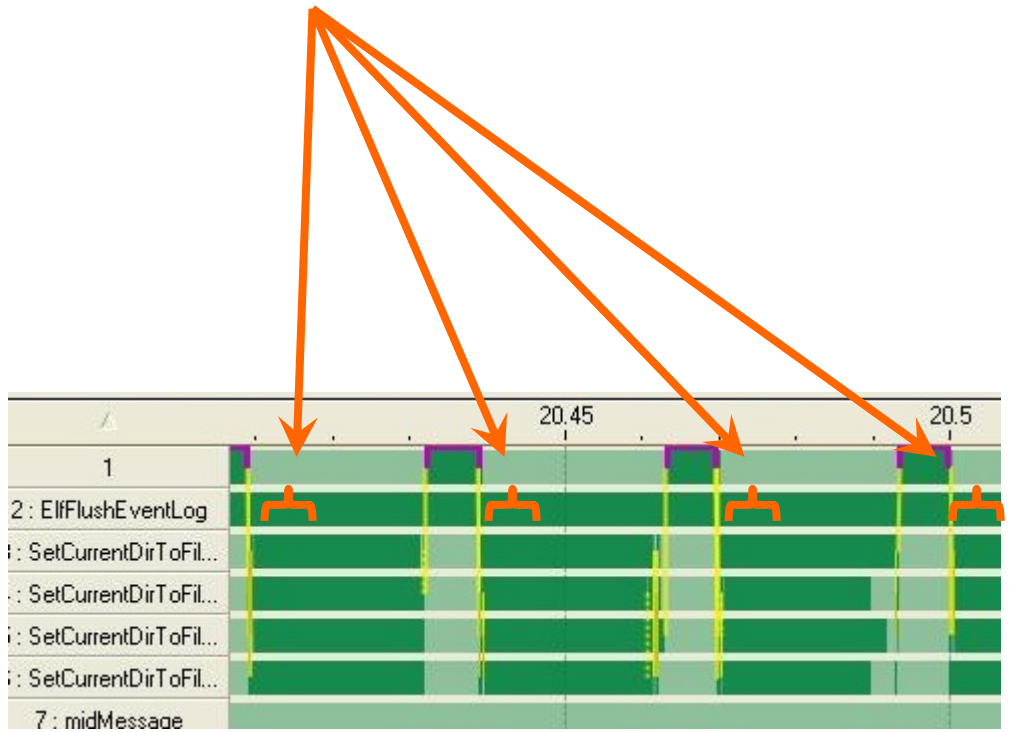


Minimizing serialization is a challenge

Heavy contention on global sync object within the Change Control Manager

Before optimization

Per-thread sync within the Change Control Manager

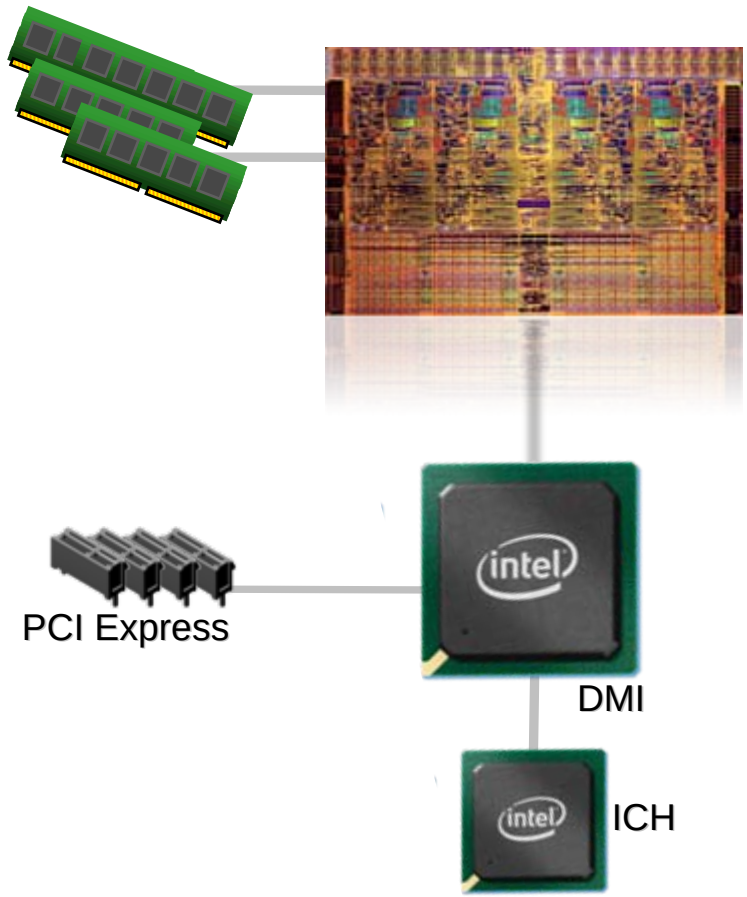


After optimization

Use these techniques to make your game really scale

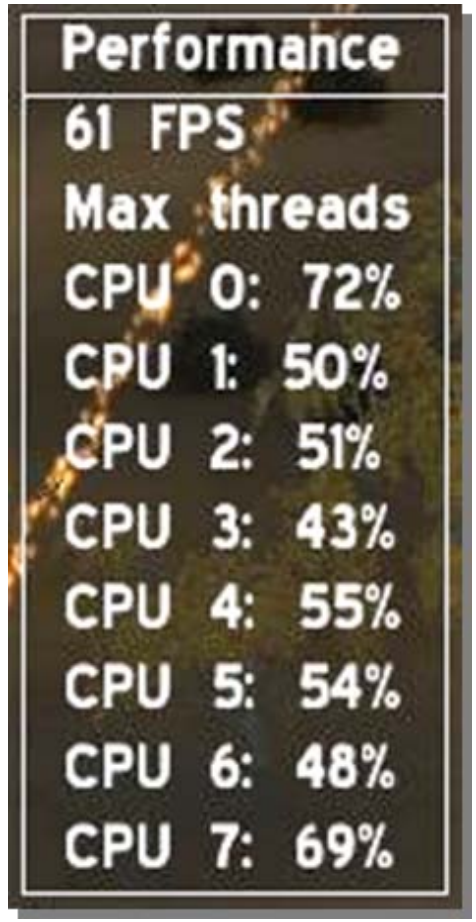
Nehalem – Overview

- Unified Cache
- Hyper-Threading Technology
- SSE 4.2
- Intel® QuickPath Architecture
- DDR3 w/ 3 Channels



Nehalem – Smoke

- More threads!! The easiest way to get a performance increase is to take advantage of all cores.
- Use a thread pool if possible. Creating threads is not free.
- Find a natural granularity. This is important to keep in mind with middleware.
- Smoke was created for Nehalem and future processors by targeting N-threads.



Nehalem – Best Practices

- Hyper-Threading just means more threads. The easiest way to unlock Nehalem's performance is threading.

- Load balance is important. Threads on a single core have their own registers but share execution resources.

- Don't use spin locks. Threads that are spinning will just waste resources useable by other threads. Use critical sections and mutexes.

- Keep cache coherency in mind. Threads should work on blocks of data instead of interleaved data.

Helpful tips

- Do:
  - Keep systems modular to help agile development. We swapped out three different physics systems with minimal impact to development deadlines.
  - Don't use spin locks.
  - Find the best task granularity for each system.
  - If possible, support multiple threading subsystems to simplify debugging.

- Don't:
  - Don't ignore thread interaction between systems, especially middleware.
  - Don't Panic. No one method works for everybody.

Techniques to maximize performance on Nehalem

Summary

- Threading games is hard, but worthwhile
- Smoke can show you how to thread effectively
- Use these techniques to make your game really scale
- Techniques to maximize performance on Nehalem

Call to Action

- Contact us at SmokeCode@Intel.com and we'll keep you up-to-date
- For more info on Nehalem and other Intel products visit: softwarecommunity.intel.com

Additional sources of information on this topic:

Sessions:

- Intel® Larrabee: A Many-Core x86 Architecture for Visual Computing [Tue 9:45-10:35]
- Game Threading Strategies for Next Generation Intel® Microarchitecture (Nehalem) Based Platforms [Tue 10:45-11:35]
- Tools for Optimizing Visual Computing [Tue 3:00-3:50]
- Enhancing C/C++ for the Next Wave in Throughput and Visual Computing [Tue 4:00-4:50]
- Developing Connected Visual Computing Experiences using Open Source Software [Tue 5:00-5:50]
- Enhancing the Media Experience with Intel® Integrated Graphics [Wed 1:40-2:30]
- DisplayPort: Foundation and Innovation for Current and Future Intel® Platforms [Wed 2:40-3:30]

Chalk Talks:

- Intel® Visual Computing Trends [Wed – 4:00-4:50]
- Intel® Visual Computing Technology: Media and High Definition Playback [Thur 10:10-11:00]
- Intel® Visual Computing Technology: Display Subsystem [Thur 11:10-noon]

Community:

- Display Port Community

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.
- Nehalem and other code names featured are used internally within Intel to identify products that are in development and not yet publicly announced for release. Customers, licensees and other third parties are not authorized by Intel to use code names in advertising, promotion or marketing of any product or services and any such use of Intel's internal code names is at the sole risk of the user
- 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, Intel Inside, 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.
-

Risk Factors

This presentation contains forward-looking statements that involve a number of risks and uncertainties. These statements do not reflect the potential impact of any mergers, acquisitions, divestitures, investments or other similar transactions that may be completed in the future. The information presented is accurate only as of today's date and will not be updated. In addition to any factors discussed in the presentation, the important factors that could cause actual results to differ materially include the following: Demand could be different from Intel's expectations due to factors including changes in business and economic conditions, including conditions in the credit market that could affect consumer confidence; 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's results could be affected by the timing of closing of acquisitions and divestitures. 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. 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; Intel's ability to respond quickly to technological developments and to incorporate new features into its products; and the availability of sufficient supply of components from suppliers to meet demand. The gross margin percentage could vary significantly from expectations based on changes in revenue levels; product mix and pricing; capacity utilization; 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; and the timing and execution of the manufacturing ramp and associated costs, including start-up costs. Expenses, particularly certain marketing and compensation expenses, vary depending on the level of demand for Intel's products, the level of revenue and profits, and impairments of long-lived assets. Intel is in the midst of a structure and efficiency program that is resulting in several actions that could have an impact on expected expense levels and gross margin. Intel's results could be impacted by adverse economic, social, political and physical/infrastructure conditions in the countries in which 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 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. A detailed discussion of these and other factors that could affect Intel's results is included in Intel's SEC filings, including the report on Form 10-Q for the quarter ended June 28, 2008.