HPX

High Performance ParalleX
CCT – Tech Talk Series

Hartmut Kaiser (hkaiser@cct.lsu.edu)
http://stellar.cct.lsu.edu

What's HPX ?

- Exemplar ParalleX runtime system implementation
  - Targeting conventional architectures (Linux based SMPs and clusters)
  - Currently, mainly software only implementation
  - Emphasis on
    - Functionality: finding proper architecture and API's
    - Performance: finding hotspots, contention points, reduce overheads, hide latencies
    - API: finding the minimal but complete set of required functions
    - Driven by real applications (AMR, Contact, Graphs, CFD)
- Should allow retargeting to different platforms
  - Stable basis for long term migration path of applications
  - Highly modular, allows to incorporate different policy implementations
  - First experiments with custom hardware components
- Implemented in C++
  - Utilize compiler for highly optimized implementation
  - Utilize language for best possible user experience/simplest API

Technology Demands new Response

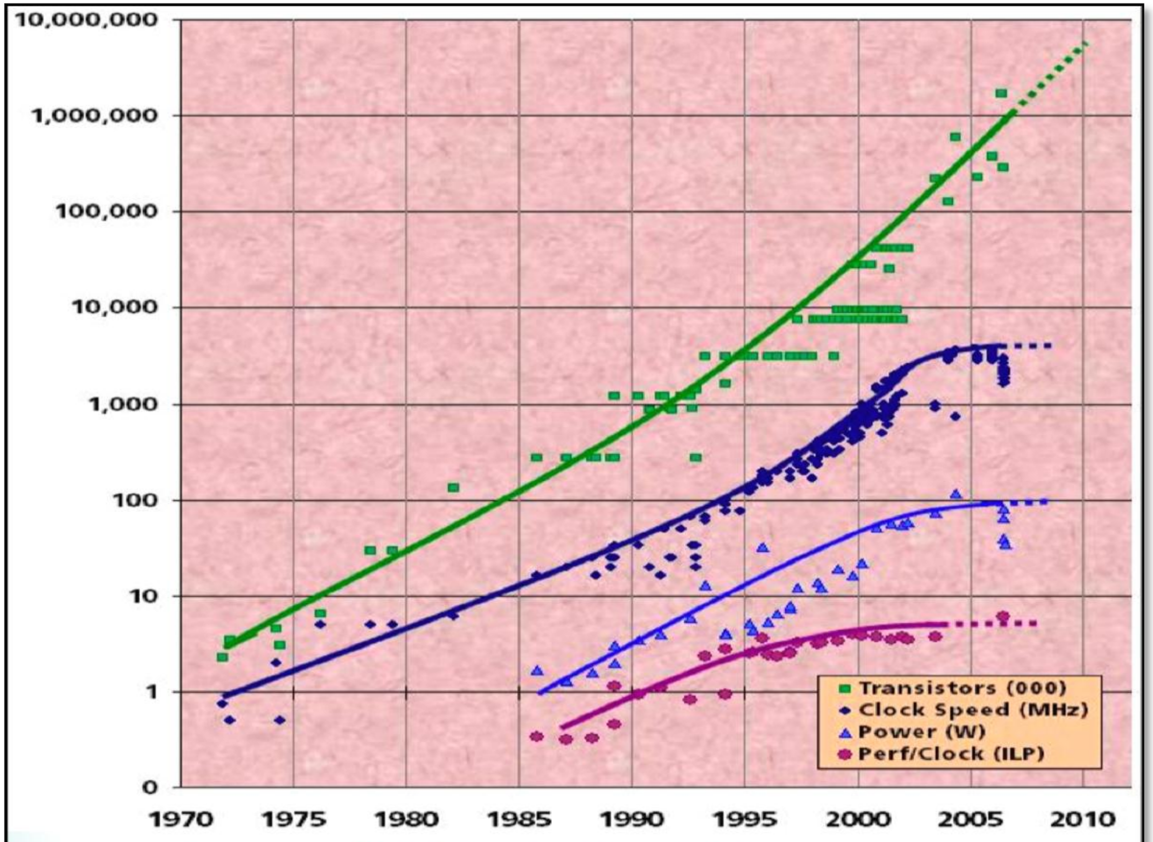


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

Technology Demands new Response

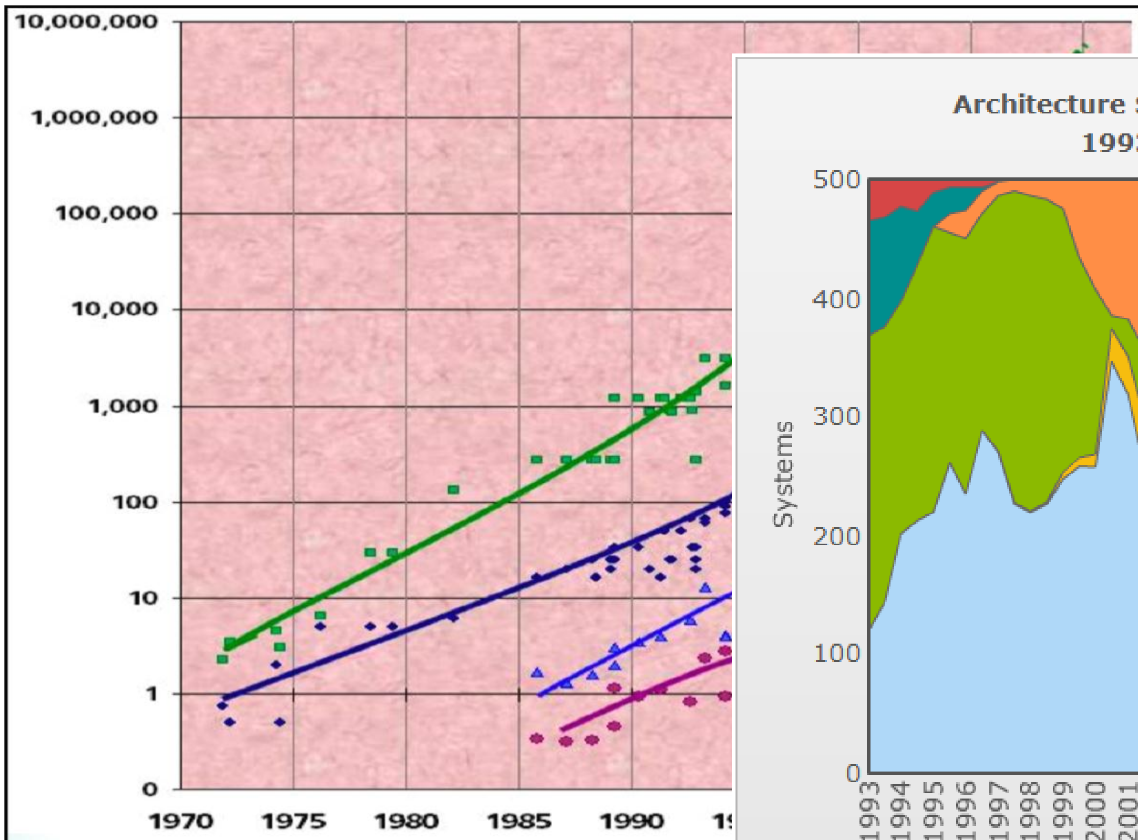
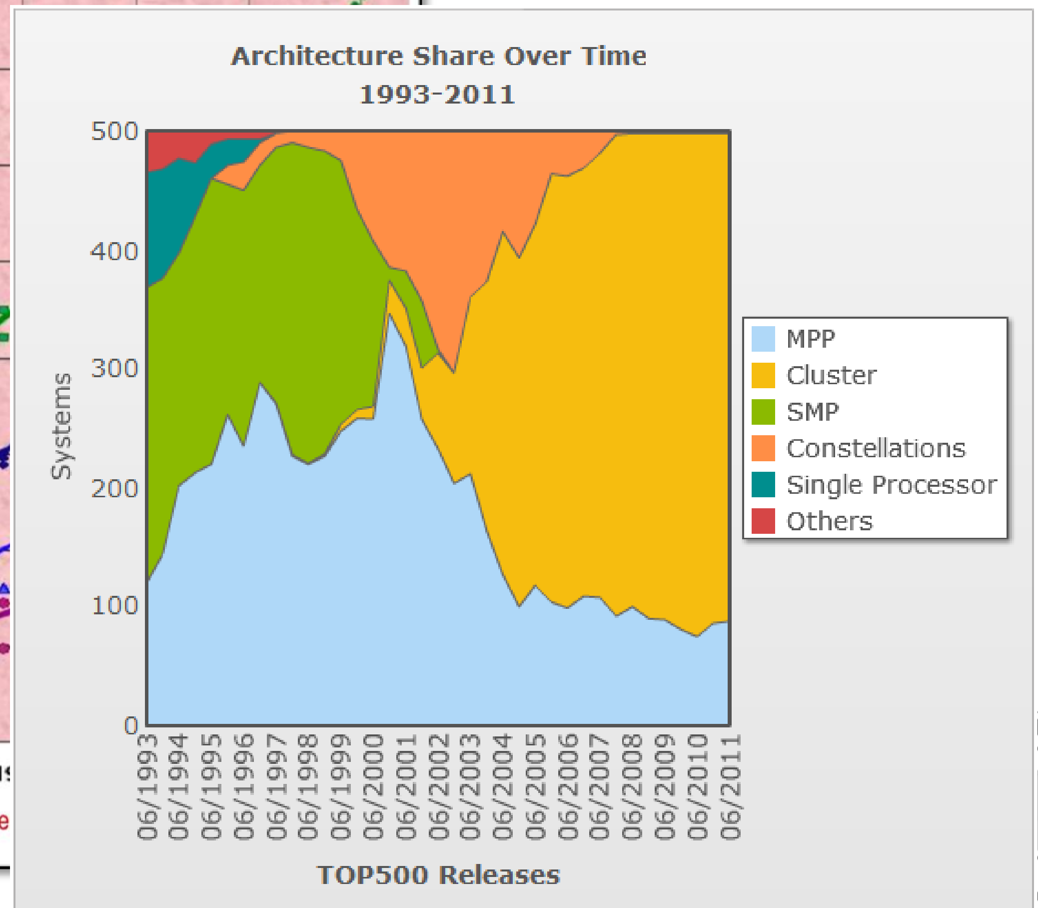


Figure courtesy of Kunle Olukotun, Lance Hammond, He
Sutter, and Burton Smith



Techno... ...onse

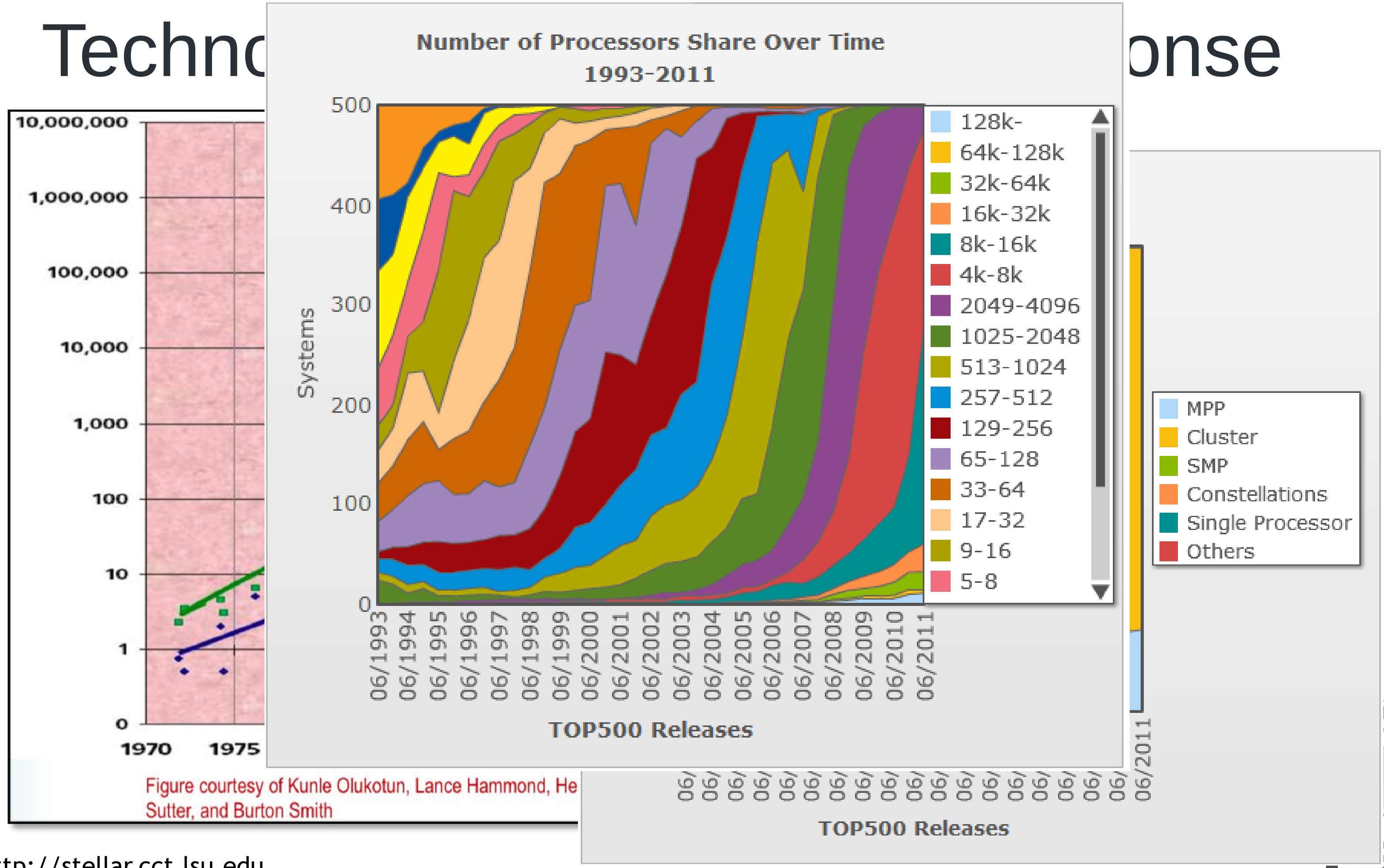


Figure courtesy of Kunle Olukotun, Lance Hammond, He... Sutter, and Burton Smith

Amdahl's Law (Strong Scaling)

$$S = \frac{1}{(1-P) + \frac{P}{N}}$$

- S: Speedup
- P: Proportion of parallel code
- N: Number of processors

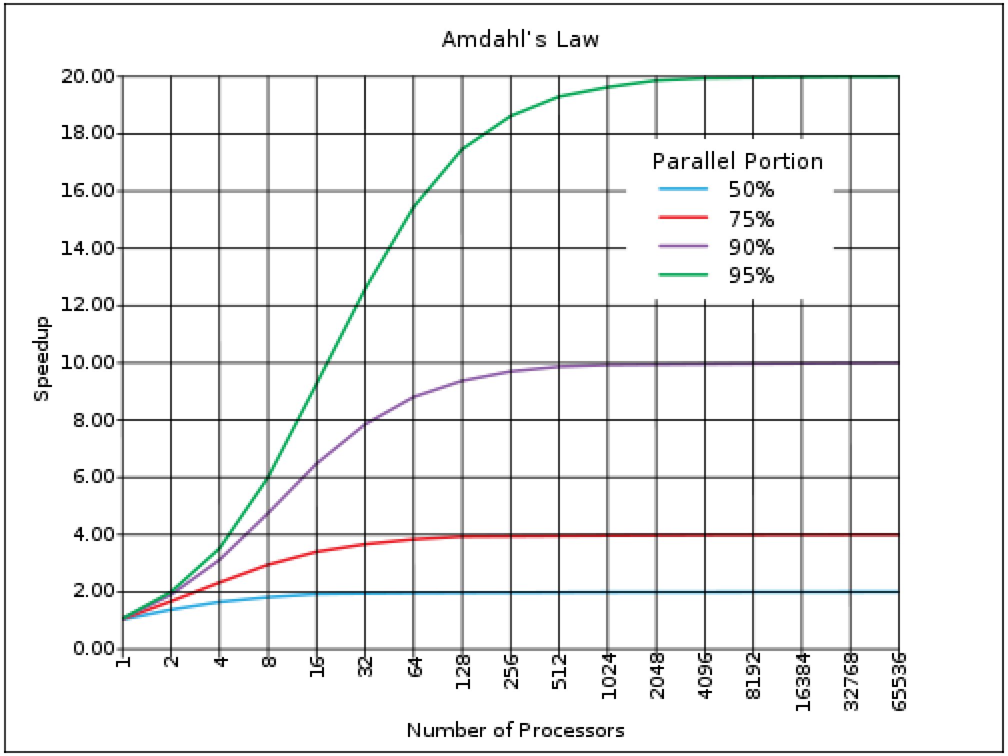


Figure courtesy of Wikipedia (http://en.wikipedia.org/wiki/Amdahl's_law)

Gustafson's Law (Weak Scaling)

$$S = N - P(N - 1)$$

- *S*: Speedup
- *P*: Proportion of parallel code
- *N*: Number of processors

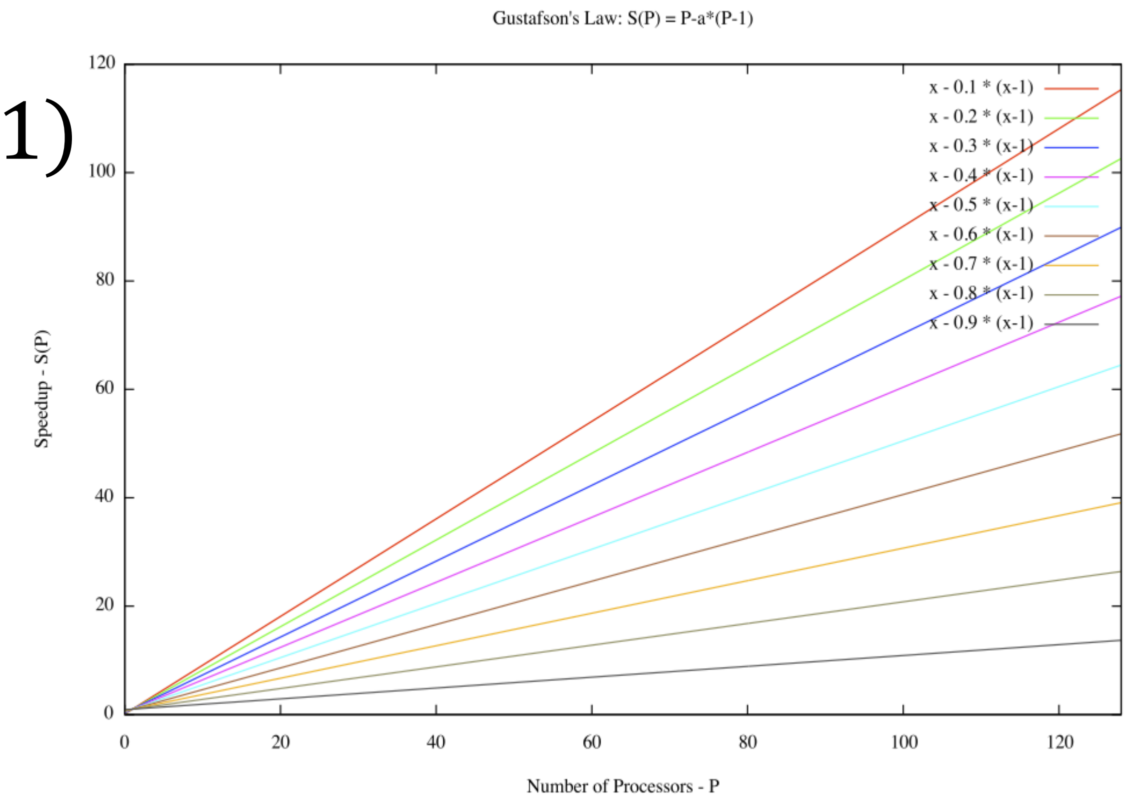


Figure courtesy of Wikipedia (http://en.wikipedia.org/wiki/Gustafson's_law)

The 4 Horsemen of the Apocalypse: SLOW

- **S**tarvation
  - Insufficient concurrent work to maintain high utilization of resources
- **L**atencies
  - Time-distance delay of remote resource access and services
- **O**verheads
  - Work for management of parallel actions and resources on critical path which are not necessary in sequential variant
- **W**aiting for Contention resolution
  - Delays due to lack of availability of oversubscribed shared resource

courtesy of www.albrecht-durer.org

The 4 Horsemen of the Apocalypse: SLOW

- **S**tarvation
  - Insufficient concurrent work to maintain high utilization of resources
- **L**atencies
  - Time-distance delay of remote … access and services
- **O**verheads
  - Work for mana… actions and … which … va…
- …
  - …ack of availability of …bed shared resource

Impose upper bound on both, weak and strong scaling


courtesy of www.albrecht-durer.org

Main HPX Runtime System Tasks

- Manage parallel execution for the application — Starvation
  - Exposing parallelism, runtime adaptive management of parallelism and resources
  - Synchronizing parallel tasks
  - Thread (task) scheduling, load balancing
- Mitigate latencies for the application — Latency
  - Latency hiding through overlap of computation and communication
  - Latency avoidance through locality management
- Reduce overhead for the application — Overhead
  - Synchronization, scheduling, load balancing, communication, context switching, memory management, address translation
- Resolve contention for the application — Contention
  - Adaptive routing, resource scheduling, load balancing
  - Localized request buffering for logical resources

What's ParalleX ?

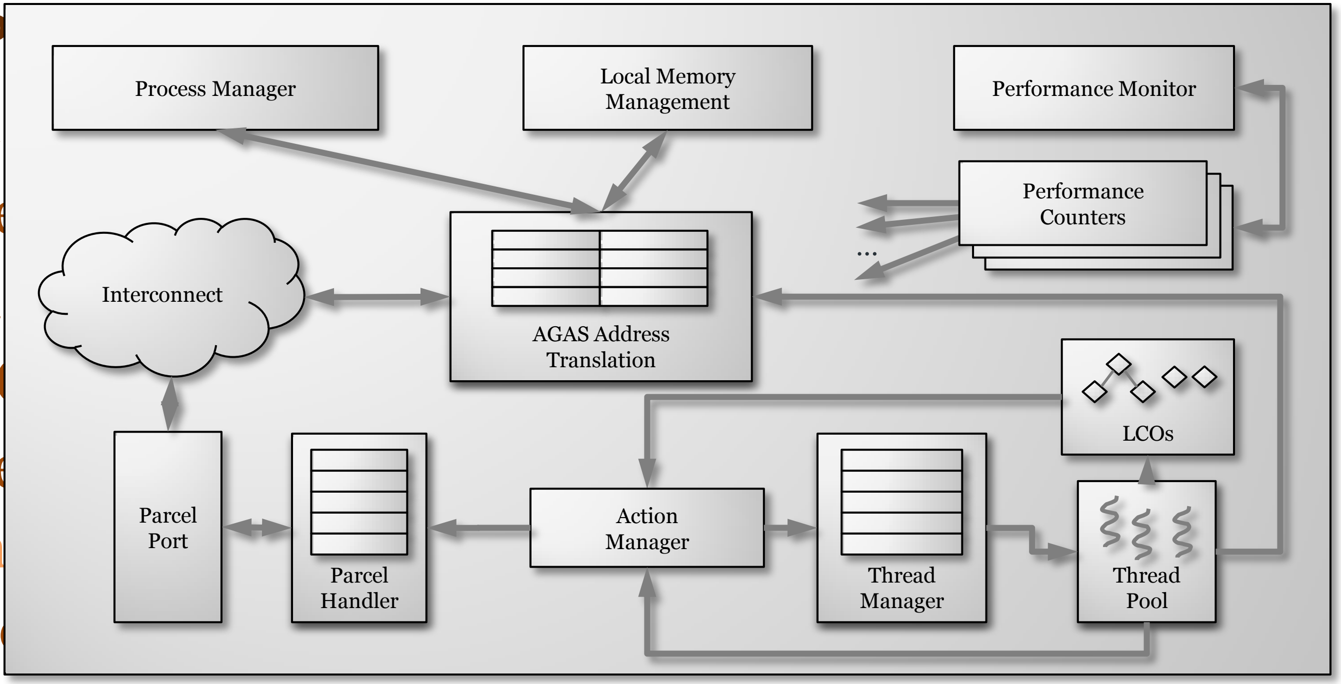
- Active global address space (AGAS) instead of PGAS
- Message driven instead of message passing
- Lightweight control objects instead of global barriers
- Latency hiding instead of latency avoidance
- Adaptive locality control instead of static data distribution
- Moving work to data instead of moving data to work
- Fine grained parallelism of lightweight threads instead of Communicating Sequential Processes (CSP/MPI)

HPX Runtime System Design

- Current version of HPX provides the following infrastructure on conventional systems as defined by the ParalleX execution model
  - Active Global Address Space (AGAS)
  - ParalleX Threads and ParalleX Thread Management
  - Parcel Transport and Parcel Management
  - Local Control Objects (LCOs)
  - ParalleX Processes
    - Namespace and policies management, locality control
  - Monitoring subsystem

HPX Runtime System Design

- Current version of HPX provides the following infrastructure on conventional systems as defined by the P
  - Active
  - Paralle
  - Parcel
  - Local
  - Paralle
    - Nam
  - Monit

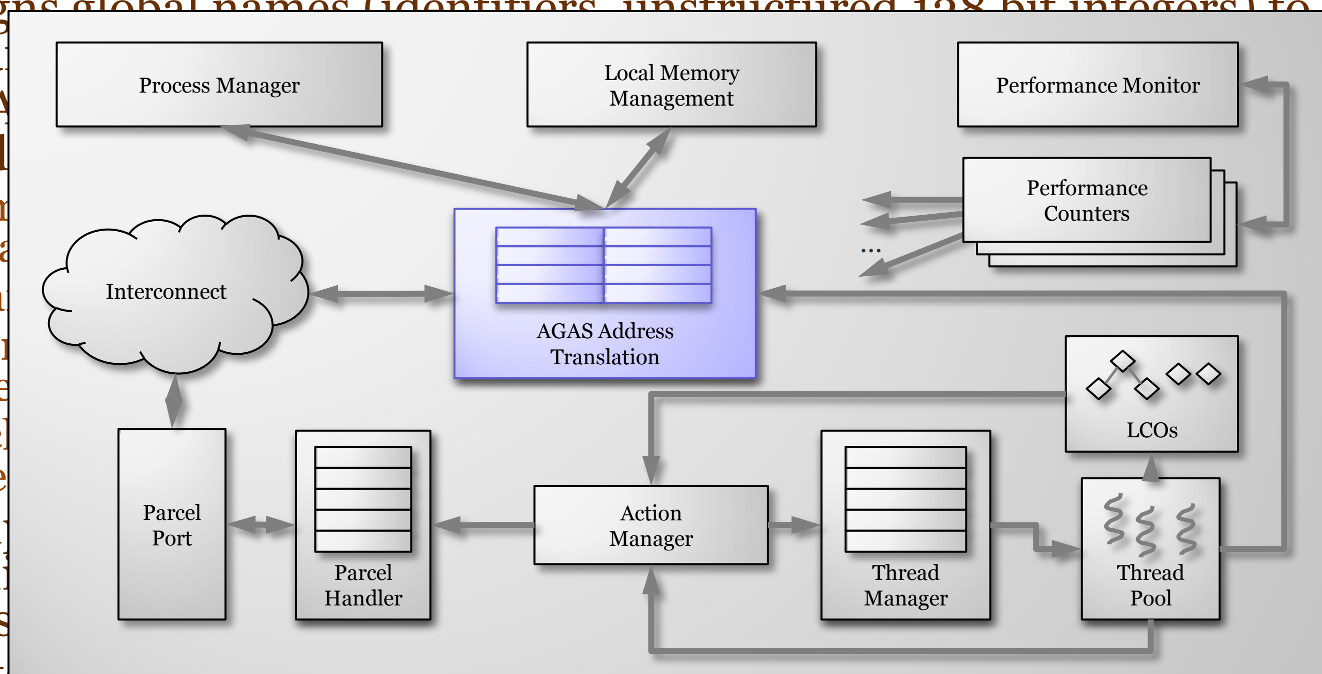


Active Global Address Space

- Global Address Space throughout the system
  - Removes dependency on static data distribution
  - Enables dynamic load balancing of application and system data
- AGAS assigns global names (identifiers, unstructured 128 bit integers) to all entities managed by HPX.
- Unlike PGAS provides a mechanism to resolve global identifiers into corresponding local virtual addresses (LVA)
  - LVAs comprise – Locality ID, Type of Entity being referred to and its local memory address
  - Moving an entity to a different locality updates this mapping
  - Current implementation is based on centralized database storing the mappings which are accessible over the local area network.
  - Local caching policies have been implemented to prevent bottlenecks and minimize the number of required round-trips.
- Current implementation allows autonomous creation of globally unique ids in the locality where the entity is initially located and supports memory pooling of similar objects to minimize overhead
- Implemented garbage collection scheme of HPX objects

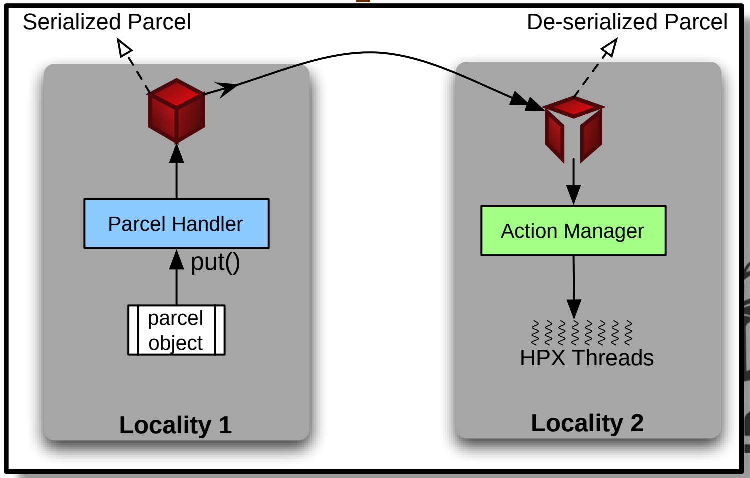
Active Global Address Space

- Global Address Space throughout the system
  - Removes dependency on static data distribution
  - Enables dynamic load balancing of application and system data
- AGAS assigns global names (identifiers, unstructured 128 bit integers) to all entities ma
- Unlike PGA correspond
  - LVAs com memory a
  - Moving a
  - Current i which are
  - Local cac minimize
- Current im s in the locali pooling of s
- Implement



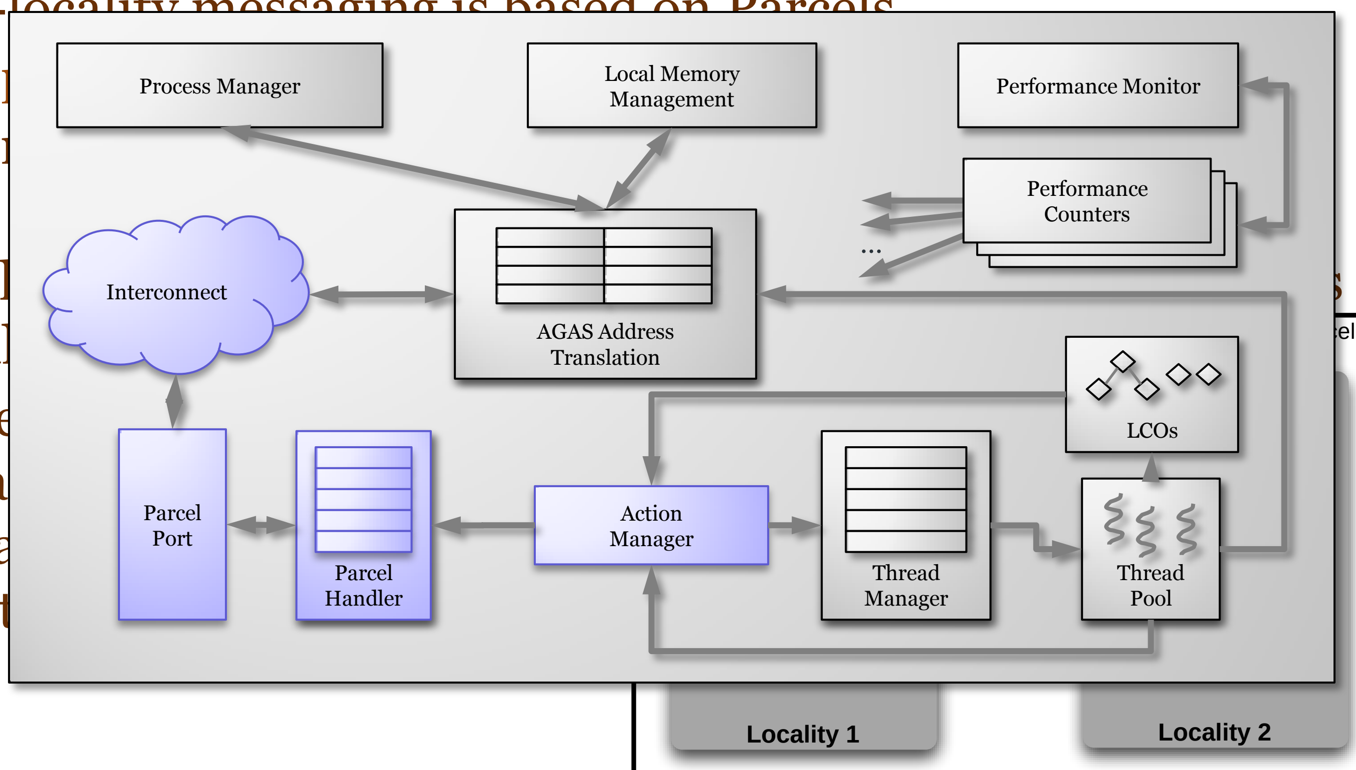
Parcel Management

- Active messages (parcels)
  - Destination address, function to execute, parameters, continuation
- Any inter-locality messaging is based on Parcels
  - In HPX parcels are represented as polymorphic objects
- An HPX entity on creating a parcel object hands it to the parcel handler.
- The parcel handler serializes the parcel where all dependent data is bundled along with the parcel
- At the receiving locality the parcel is de-serialized and causes a HPX thread to be created based on its content



Parcel Management

- Active messages (parcels)
  - Destination address, function to execute, parameters, continuation
- Any inter-locality messaging is based on Parcels
  - In HPX p
- An HPX e handler.
- The parcel bundled al
- At the rece is de-seria HPX threa its content

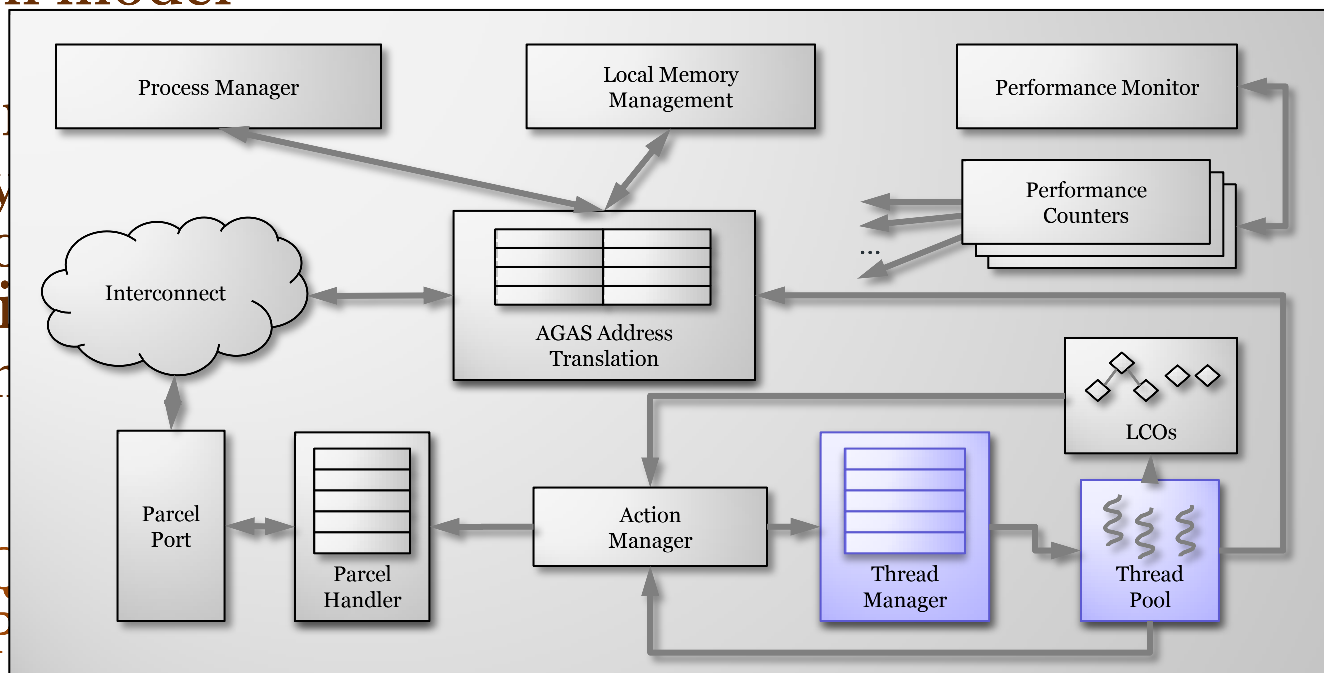


Thread Management

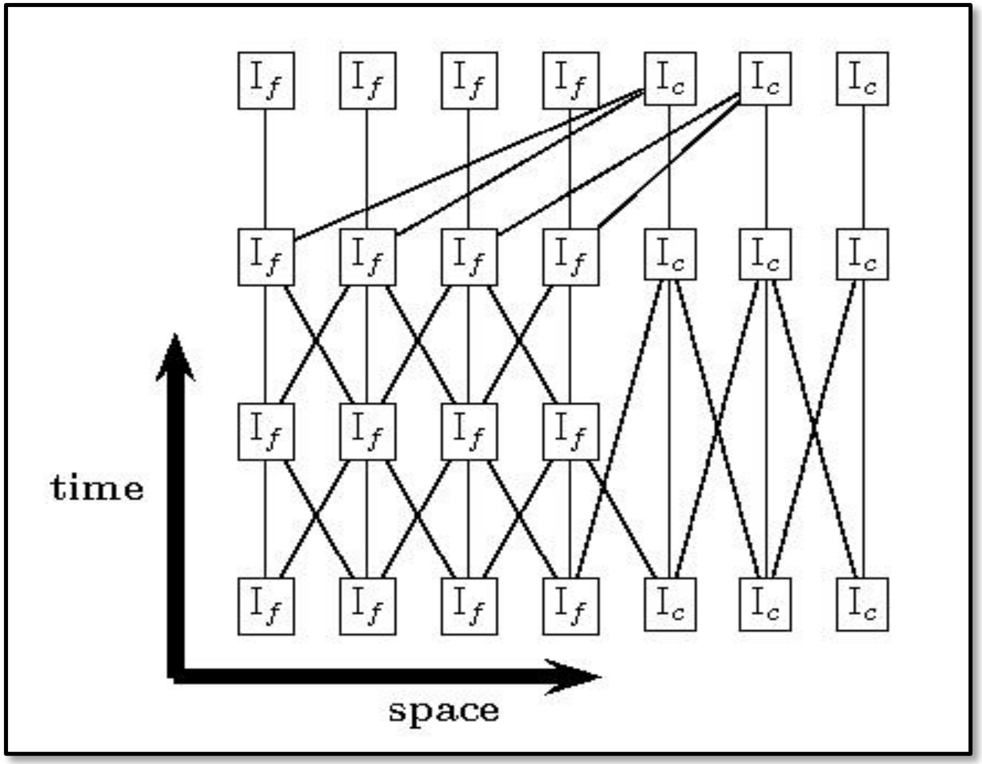
- Thread manager is modular and implements a work-queue based management as specified by the ParalleX execution model
- Threads are cooperatively scheduled at user level without requiring a kernel transition
- Specially designed synchronization primitives such as semaphores, mutexes etc. allow synchronization of PX-threads in the same way as conventional threads
- Thread management currently supports several key modes
  - Global Thread Queue
  - Local Queue (work stealing)
  - Local Priority Queue (work stealing)

Thread Management

- Thread manager is modular and implements a work-queue based management as specified by the ParalleX execution model
- Threads without
- Specially semapho threads i
- Thread r modes
  - Global
  - Local Q
  - Local P



Constraint based Synchronization



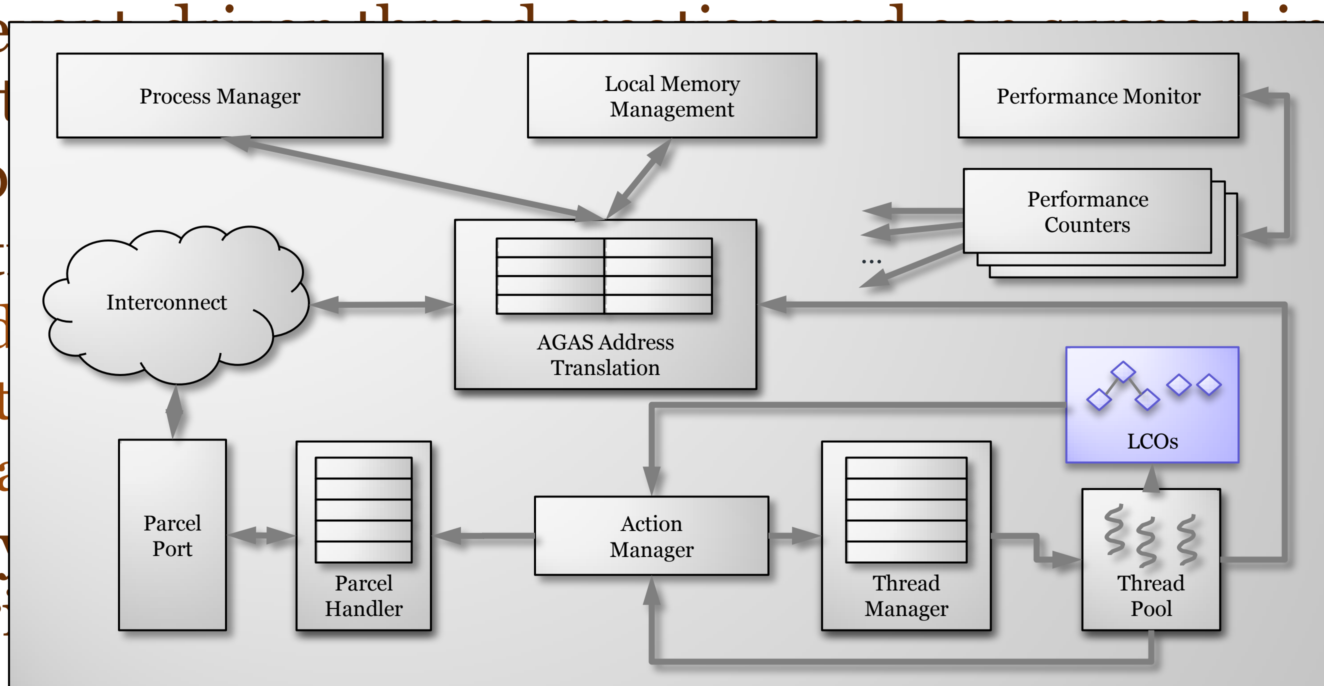
- Compute dependencies at task instantiation time
- No global barriers, uses constraint based synchronization
- Computation flows at its own pace
- Message driven
- Symmetry between local and remote task creation/execution
- Possibility to control grain size

LCOs (Local Control Objects)

- LCOs provide a means of controlling parallelization and synchronization of PX-threads
- Enable event-driven thread creation and can support in-place data structure protection and on-the-fly scheduling
- Preferably embedded in the data structures they protect
- Abstraction of a multitude of different functionalities for
  - event driven PX-thread creation,
  - protection of data structures from race conditions
  - automatic on-the-fly scheduling of work
- LCO may create (or reactivate) a PX-thread as a result of ‘being triggered’

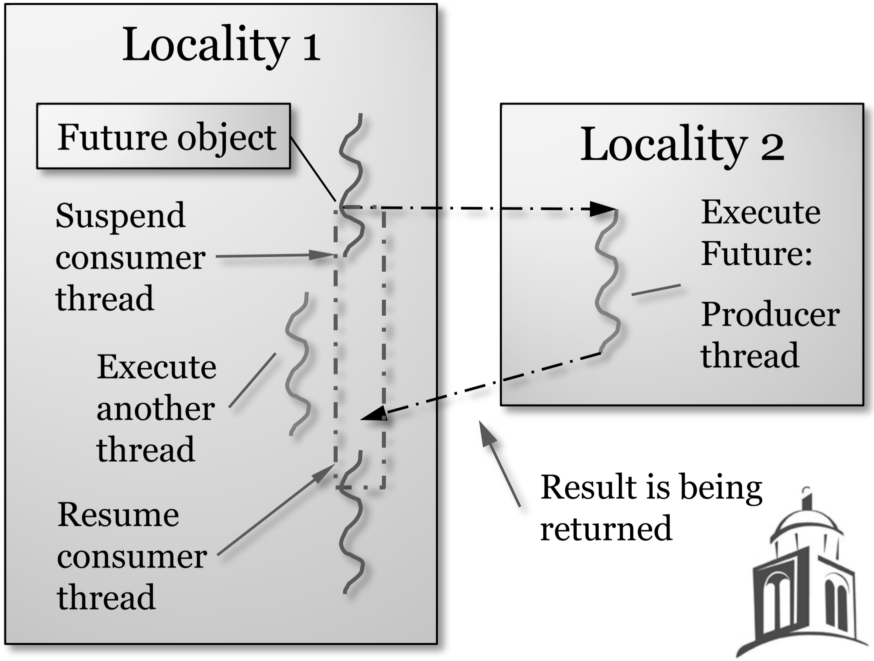
LCOs (Local Control Objects)

- LCOs provide a means of controlling parallelization and synchronization of PX-threads
- Enable e
  place dat … g
- Preferab
- Abstract … r
  - event d
  - protect
  - automa
- LCO may … f
  'being tr



Exemplar LCO: Futures

- In HPX Futures LCO refers to an object that acts as a proxy for the result that is initially not known.
- When a user code invokes a future (using future.get()) the thread can do one of 2 activities
  - If the remote data/arguments are available then the future.get() operation fetches the data and the execution of the thread continues
  - If the remote data is NOT available the thread may continue until it requires the actual value; then the thread suspends allowing other threads to continue execution. The original thread re-activates as soon as the data dependency is resolved



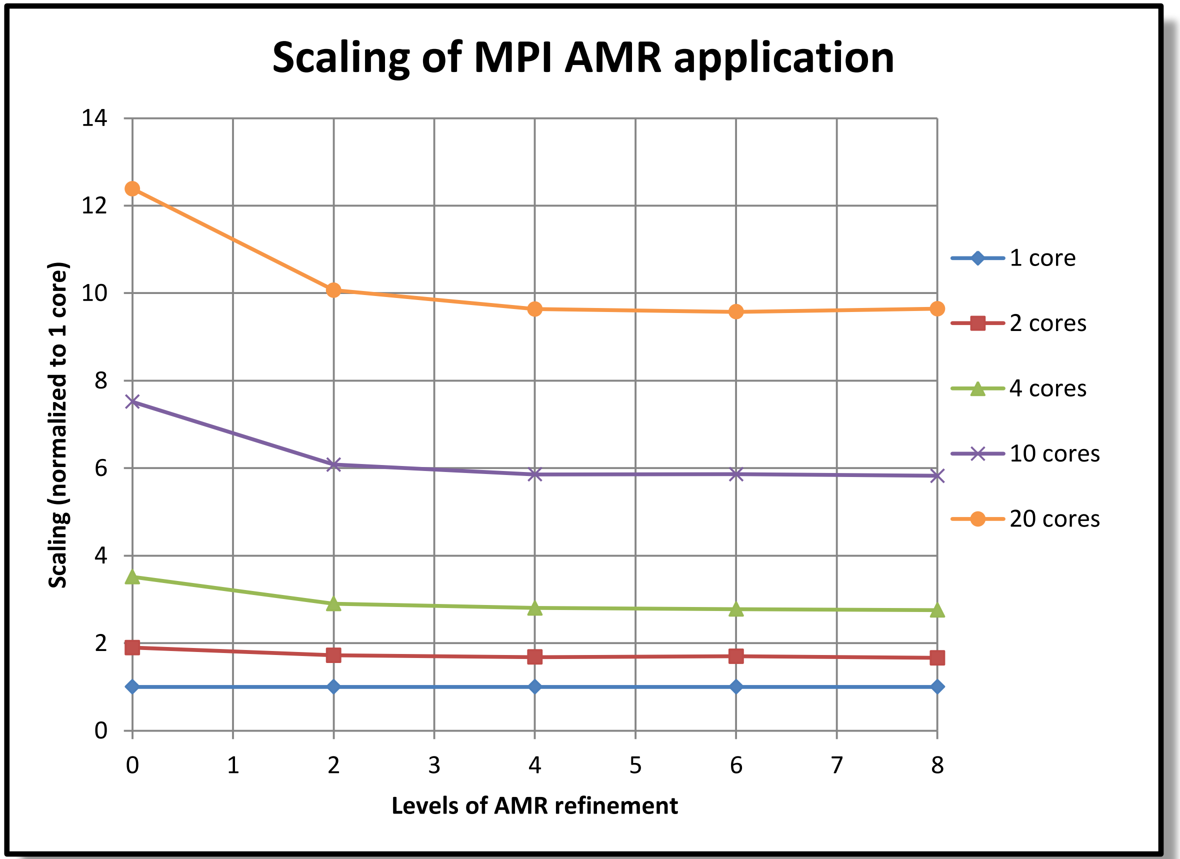
ParalleX Processes

- Management of namespace and locality
  - Centerpiece for truly distributed AGAS
  - We completed the first step in re-implementing AGAS towards being distributed
- Encapsulation of blocks of functionality and possibly distributed data
  - Completed software architecture design for processes
  - Implemented prototypical processes managing read-only distributed data

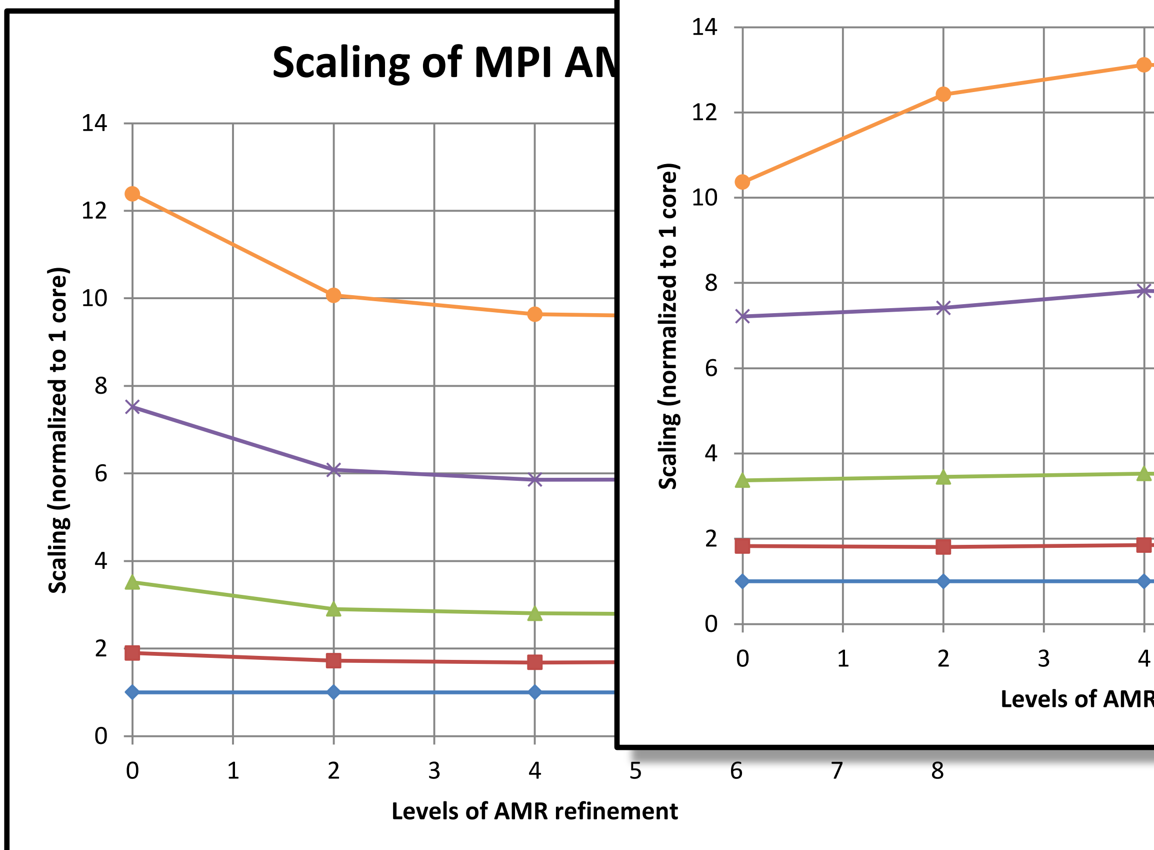
Recent Results

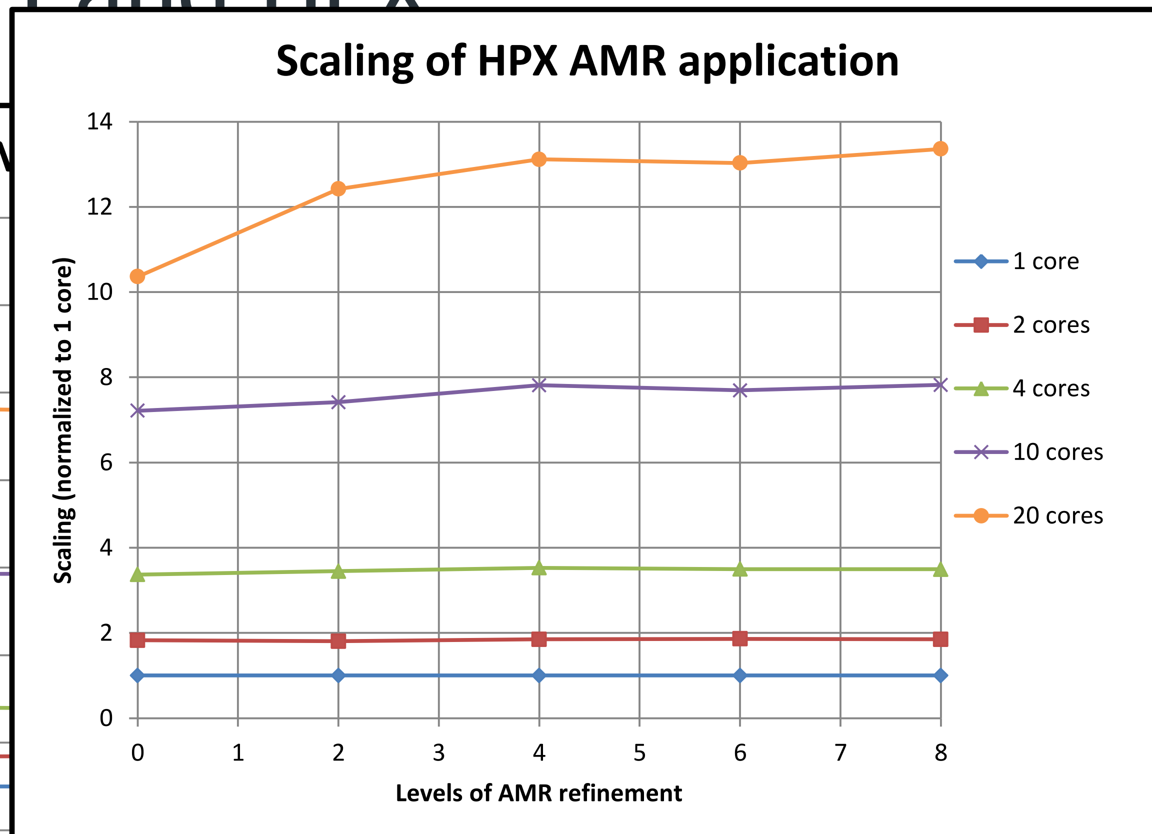
- First formal release of HPX3 (V0.6), V1.0 scheduled for May
- Re-implemented AGAS on top of Parcel transport layer
  - Better distributed scalability, better asynchrony, latency hiding
- First results implementing ParalleX processes
  - Distributed (read-only) data partitioning of very large data sets
- First (encouraging) results from distributed runs
  - Demonstrated strong scaling similar to SMP
- Consolidated performance counter monitoring framework
  - Allows to measure almost arbitrary system characteristics using unified interface
- Implemented remote application steering
  - Used to demonstrate control of power consumption
- New applications
  - Linear algebra, N-body, chess, contact, graph500

Scaling & performance: AMR using MPI and HPX

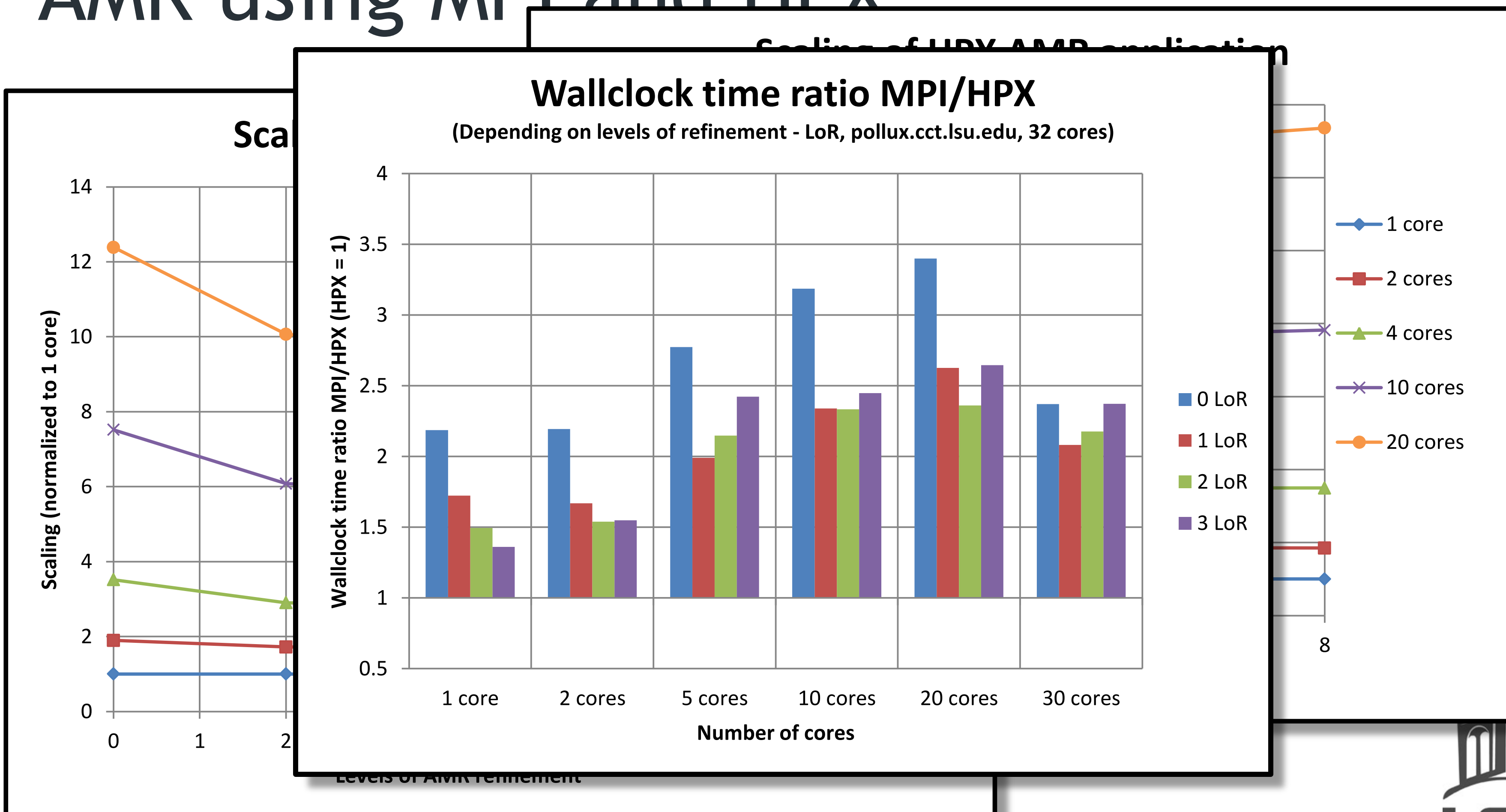


Scaling & performance: AMR using MPI and HPX

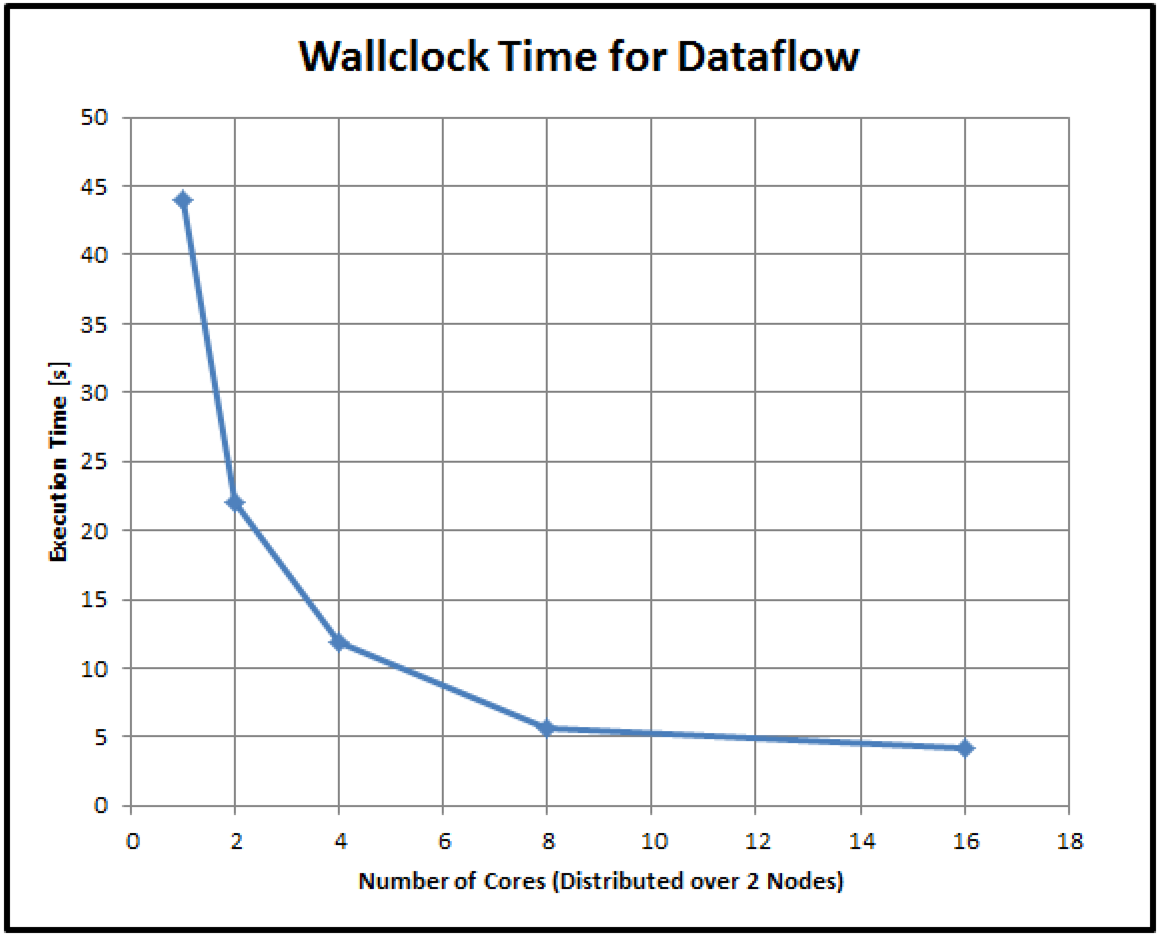




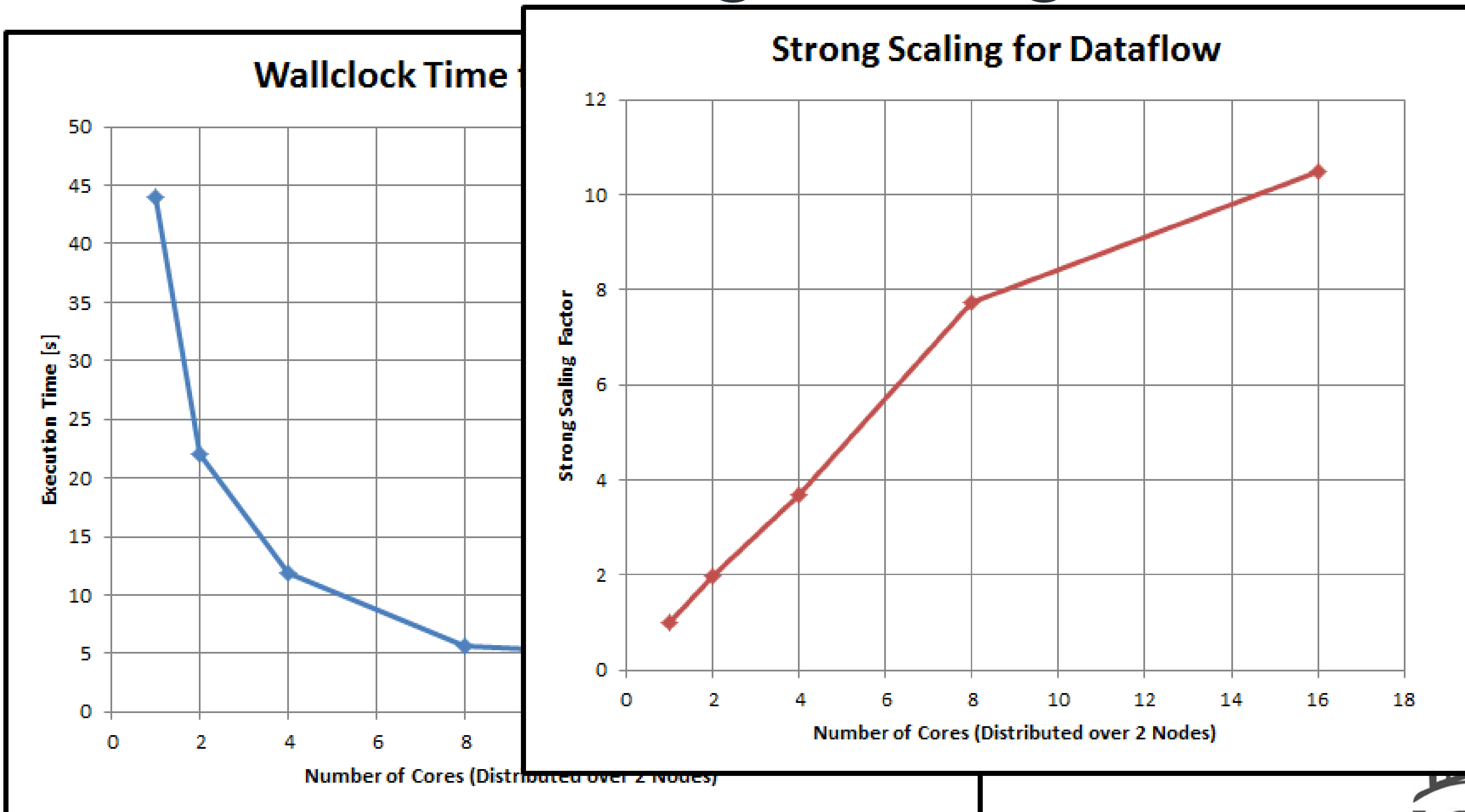
Scaling & performance: AMR using MPI and HPX



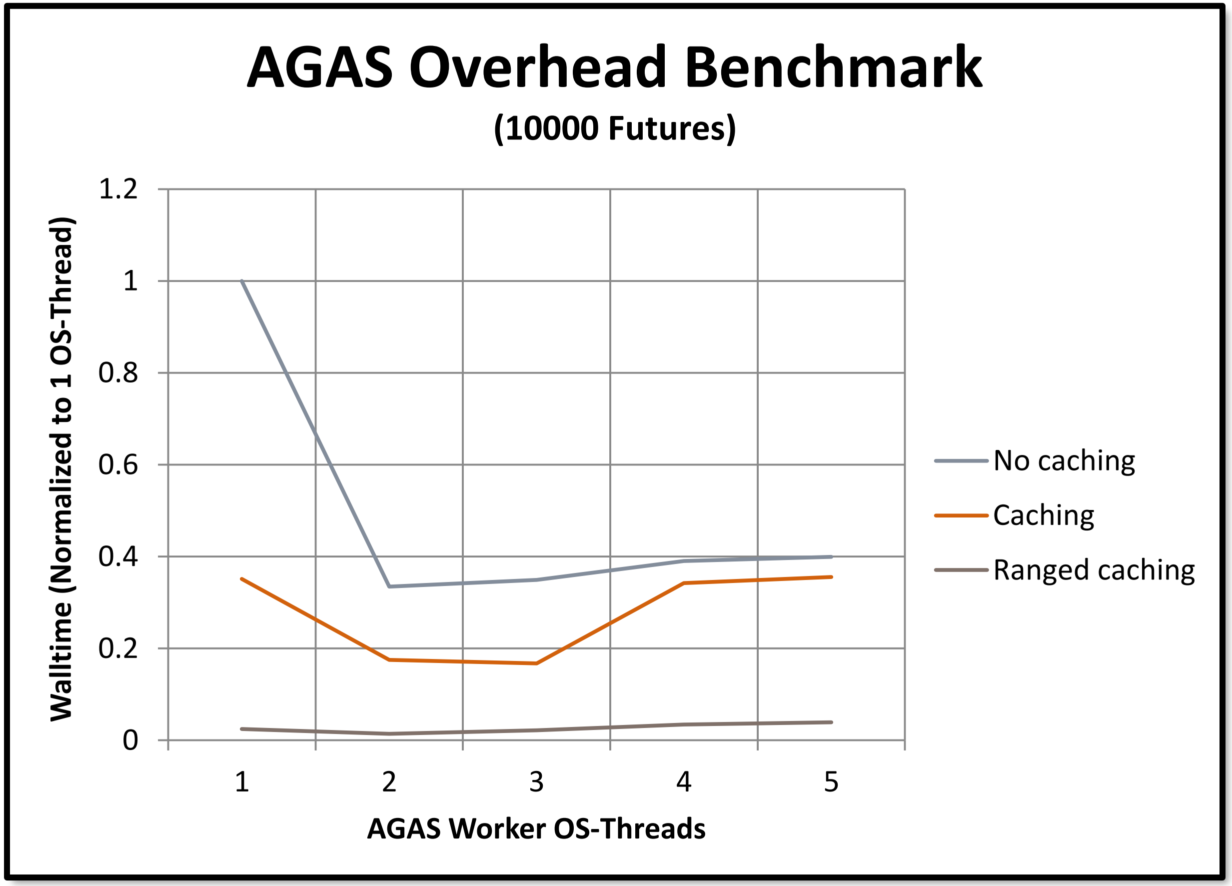
Distributed Strong Scaling



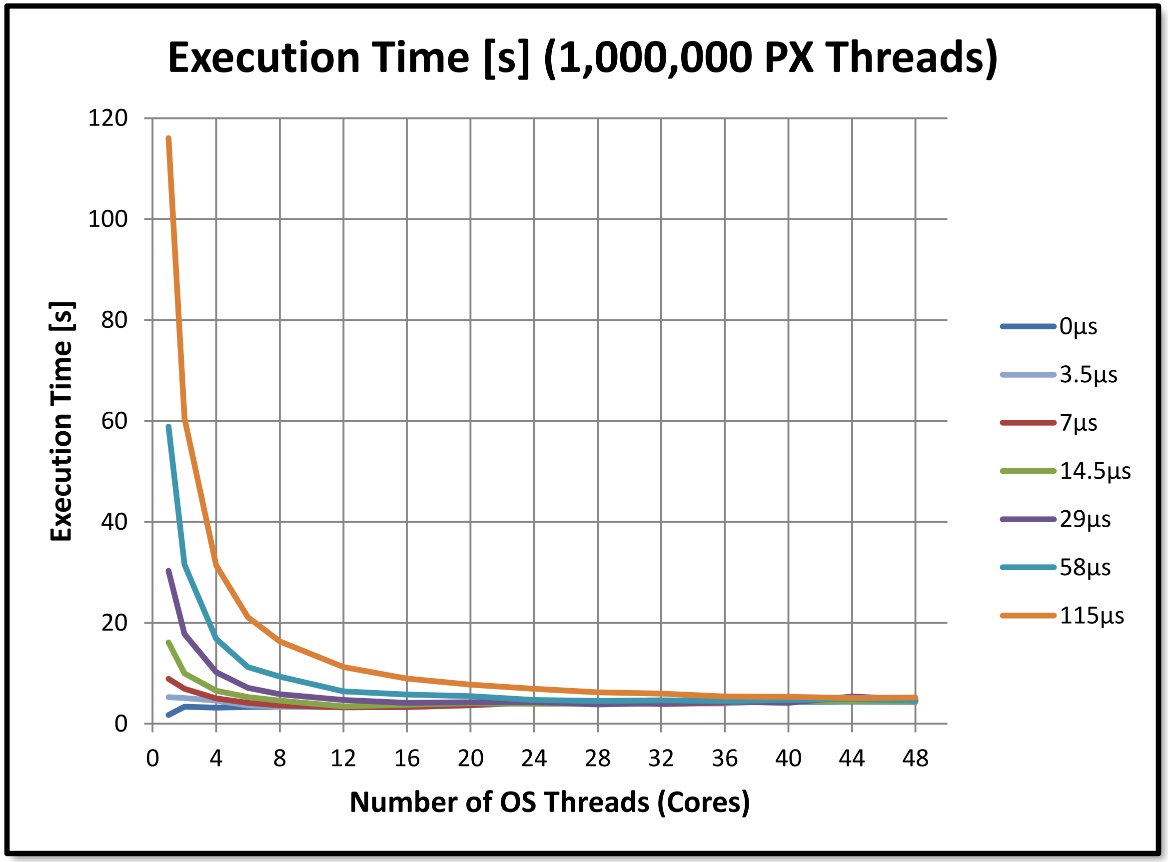
Distributed Strong Scaling



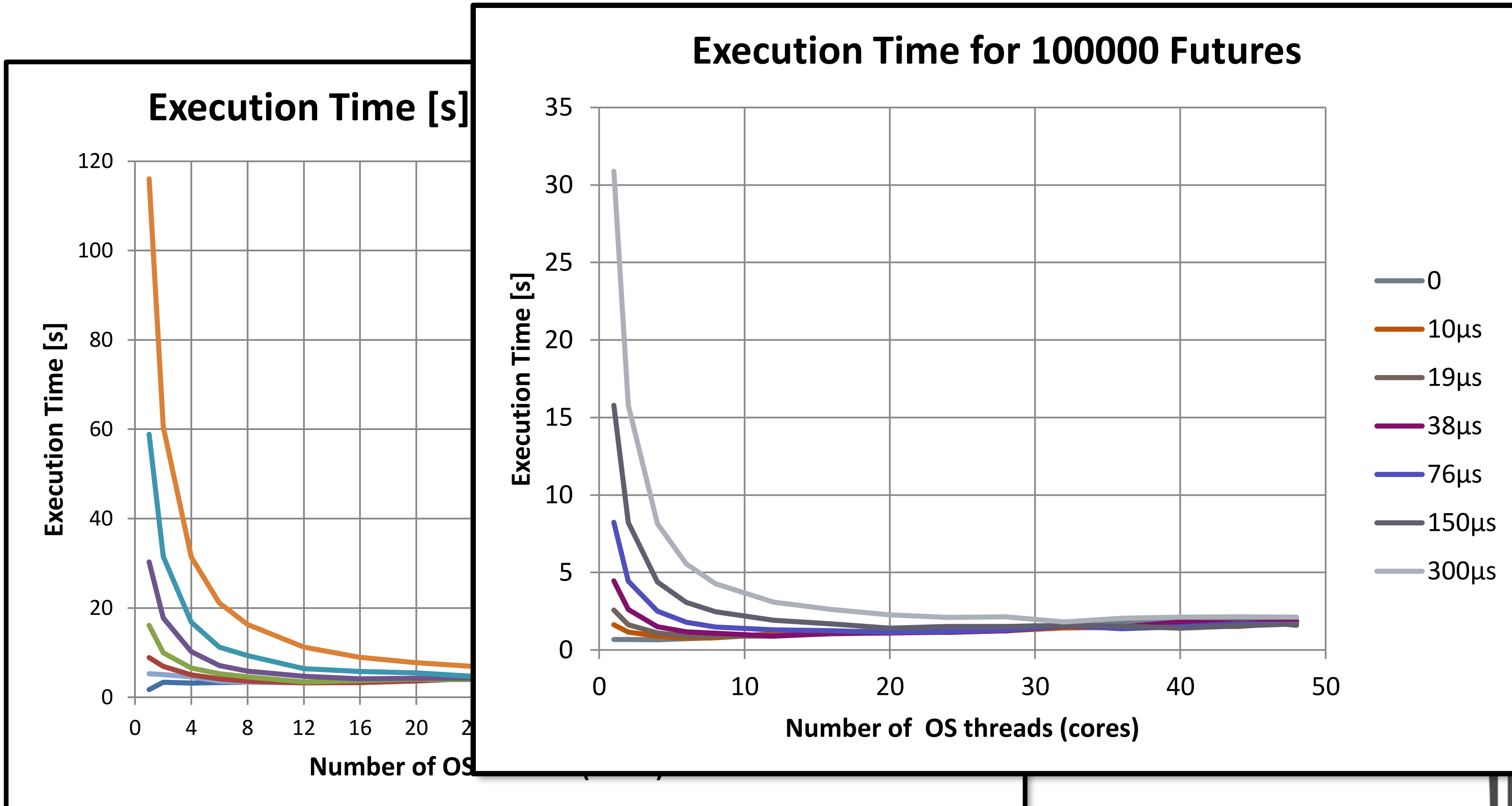
Overheads: AGAS



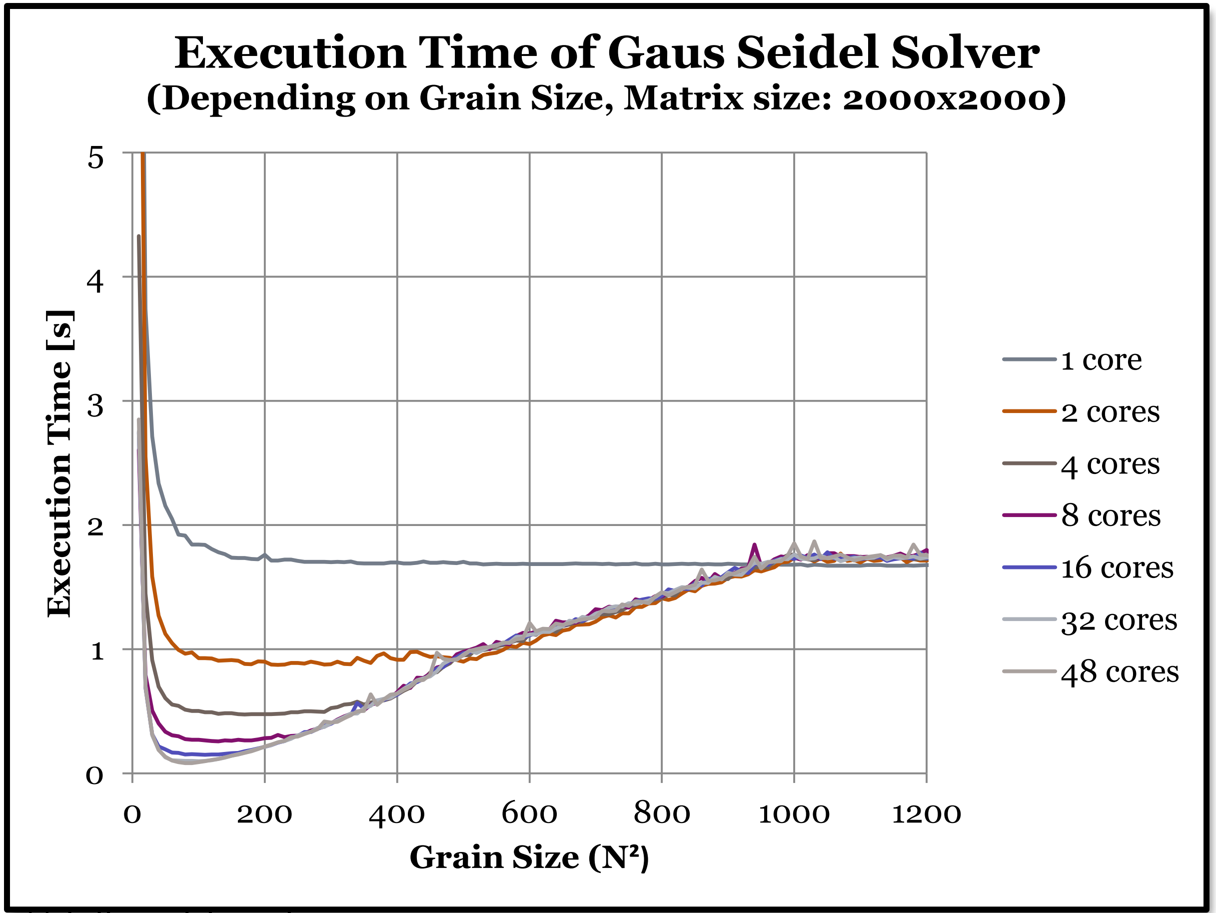
Overhead: Threads



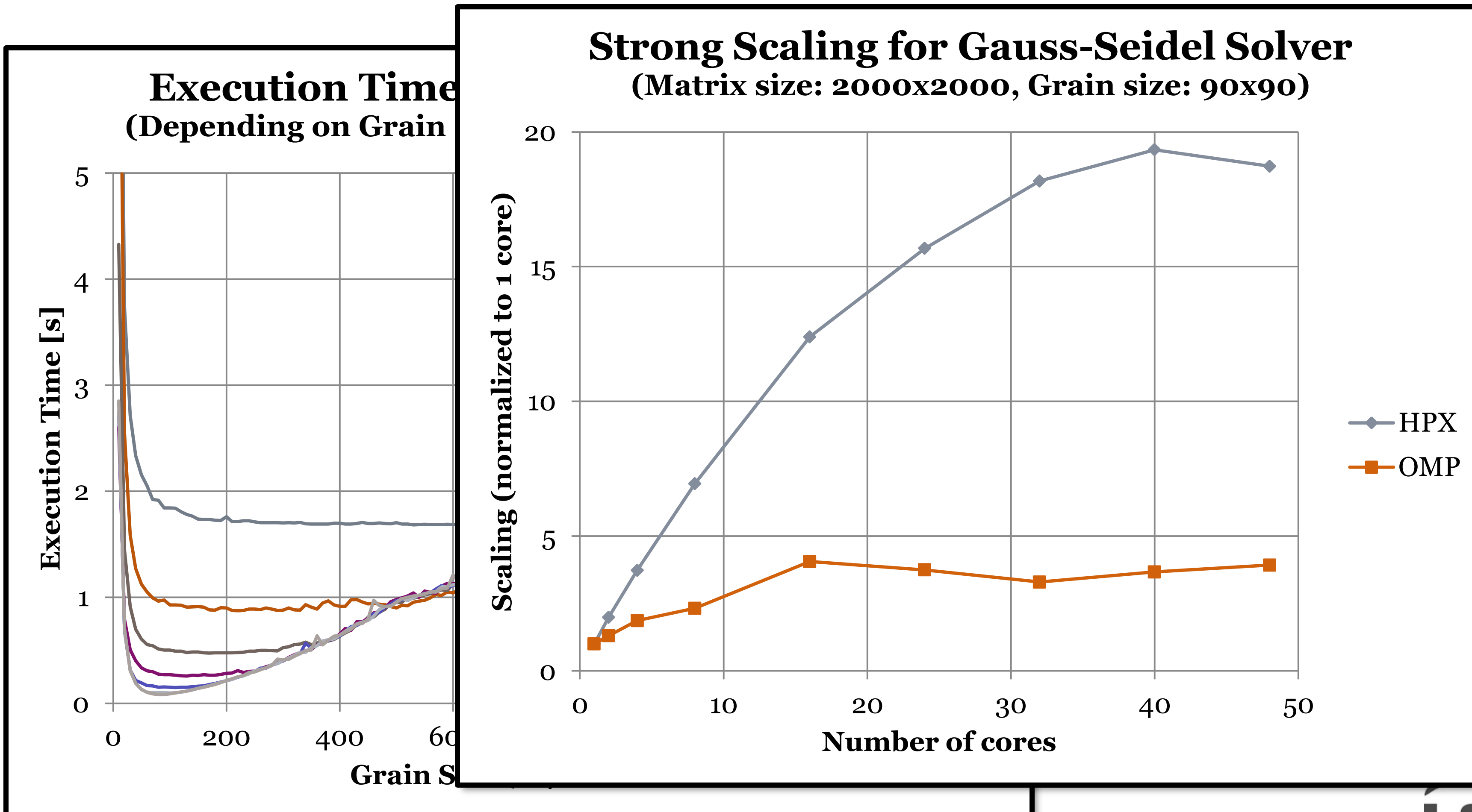
Overhead: Threads



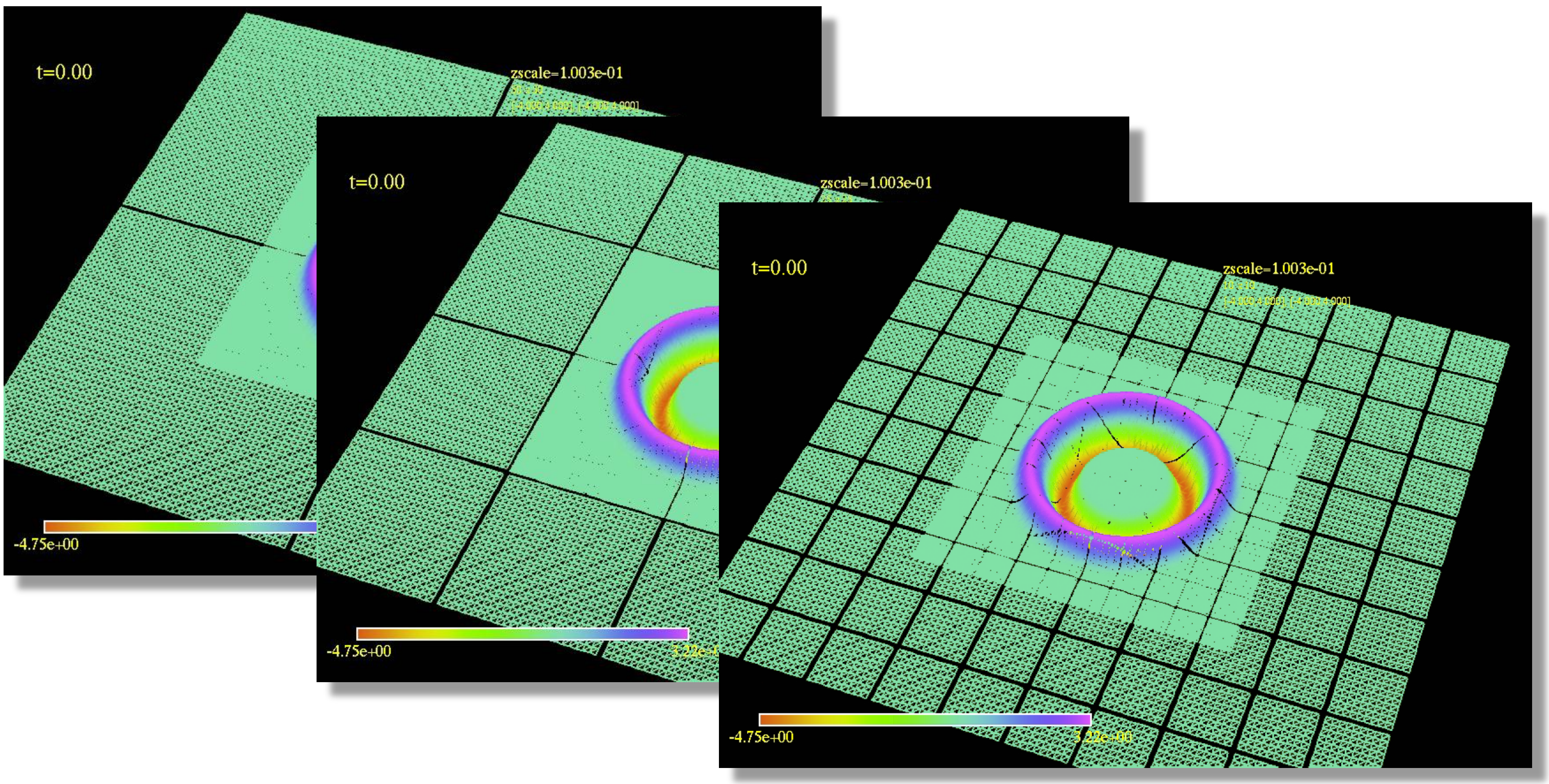
Results from a Gauss-Seidel Solver



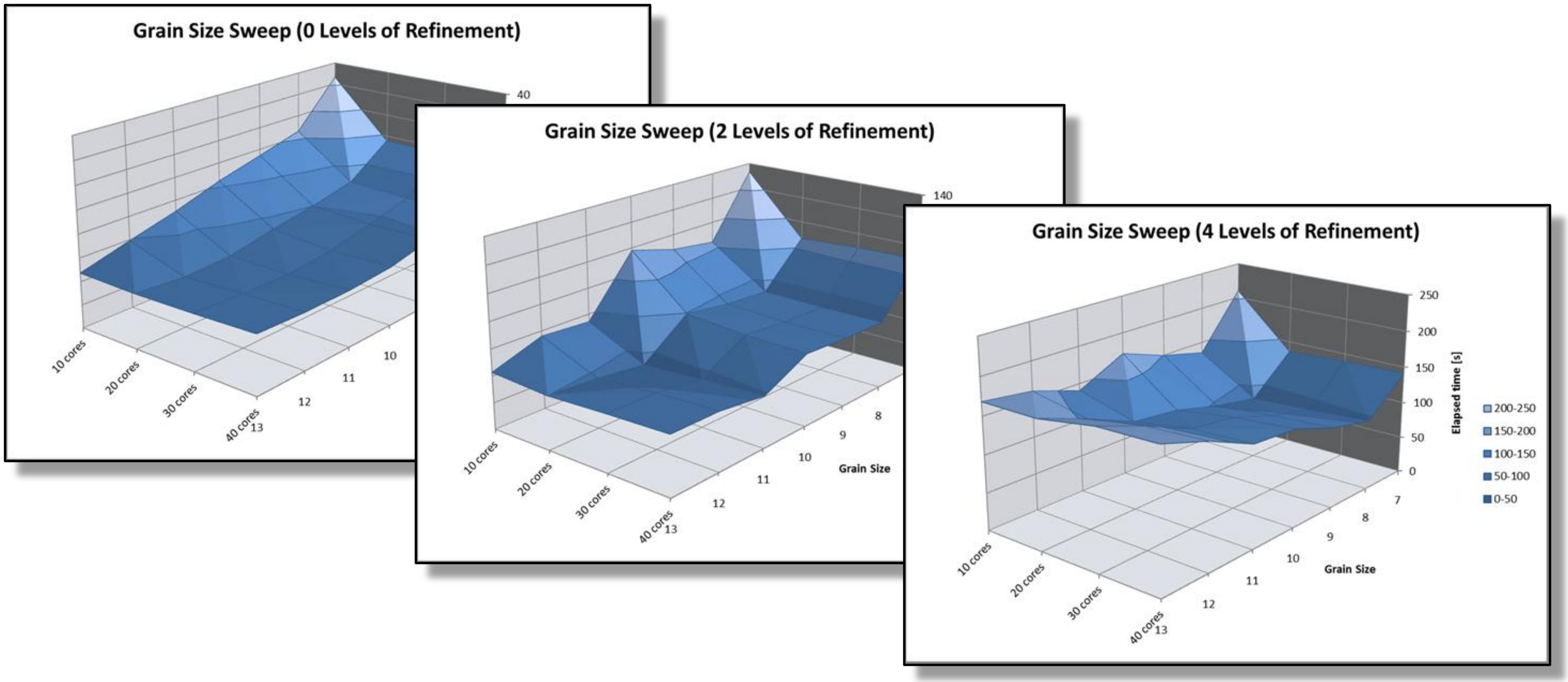
Results from a Gauss-Seidel Solver



Grain Size: The New Freedom

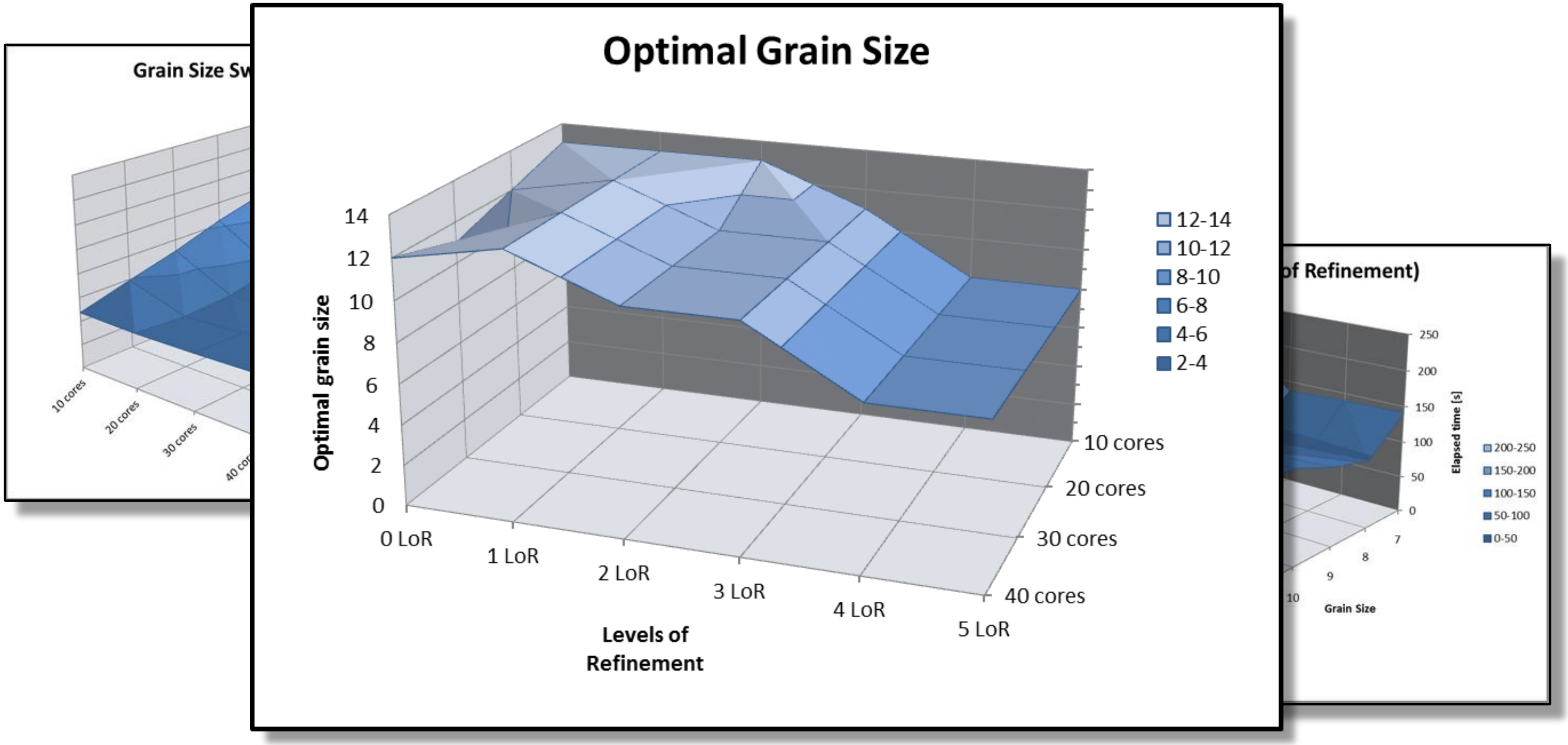


Overhead: Load Balancing



Competing effects for optimal grain size: overheads vs. load balancing (starvation)

Overhead: Load Balancing



Competing effects for optimal grain size: overheads vs. load balancing (starvation)

Conclusions

- Are we there yet?
  - Definitely NO!
  - But very promising results supporting our claim
- Are we on a right path?
  - Definitely YES!
  - Might not be THE right path, but it's a leap
- Do we have cure for those scaling impaired applications?
  - We're not sure yet!
  - Based on results we are optimistic