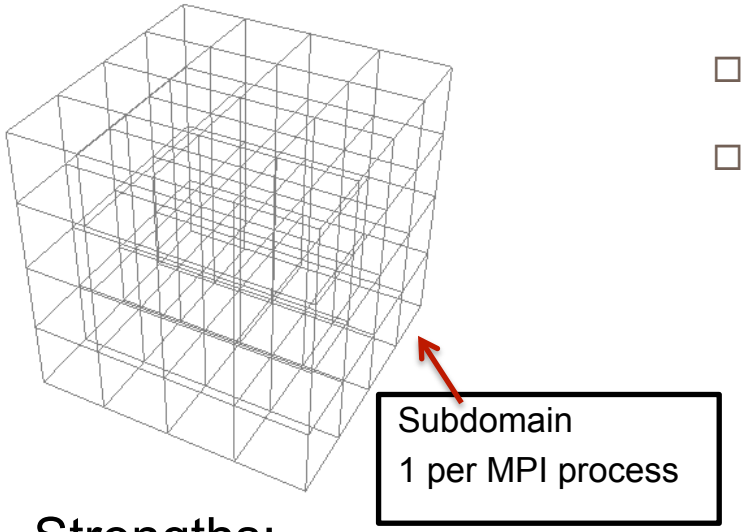
Classic HPC Application Architecture

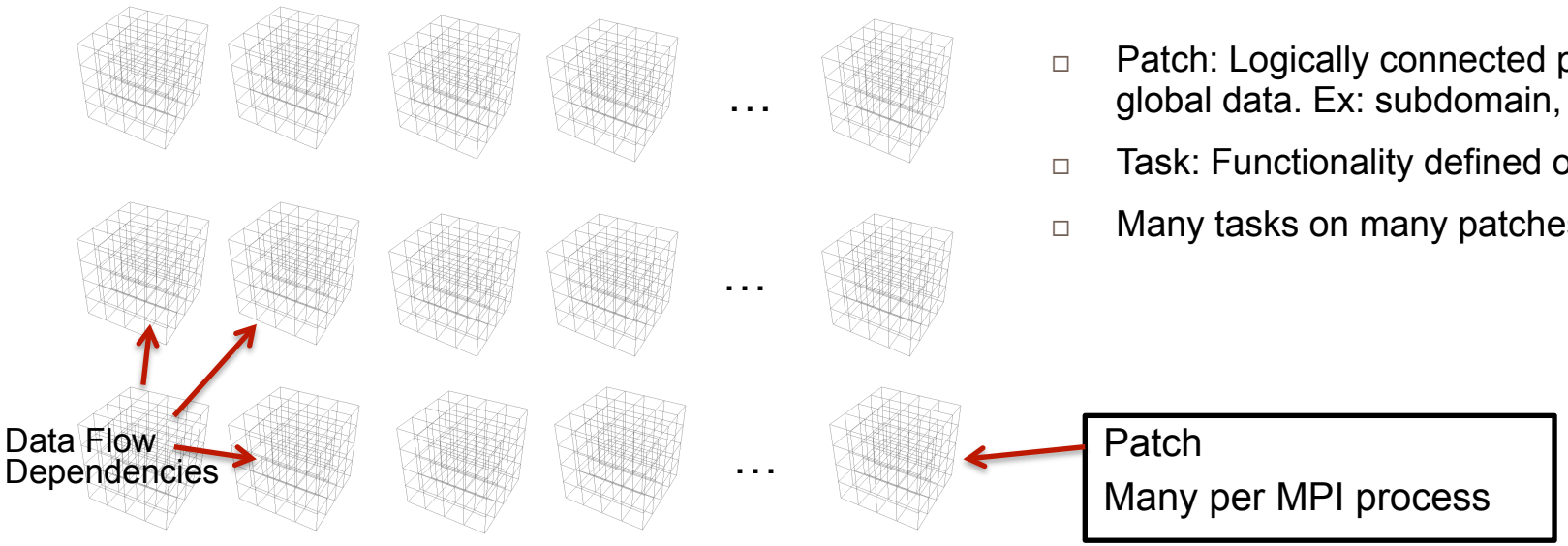


- Logically Bulk-Synchronous, SPMD
- Basic Attributes:
  - Halo exchange.
  - Local compute.
  - Global collective.
  - Halo exchange.

- Strengths:
  - Portable to many specific system architectures.
  - Separation of parallel model (SPMD) from implementation (e.g., message passing).
  - Domain scientists write sequential code within a parallel SPMD framework.
  - Supports traditional languages (Fortran, C).
  - Many more, well known.

- Weaknesses:
  - Not well suited (as-is) to emerging manycore systems.
  - Unable to exploit functional on-chip parallelism.
  - Difficult to tolerate dynamic latencies.
  - Difficult to support task/compute heterogeneity.

Task-centric/Dataflow Application Architecture



- Patch: Logically connected portion of global data. Ex: subdomain, subgraph.
- Task: Functionality defined on a patch.
- Many tasks on many patches.

- Strengths:
  - Portable to many specific system architectures.
  - Separation of parallel model from implementation.
  - Domain scientists write sequential code within a parallel framework.
  - Supports traditional languages (Fortran, C).
  - Similar to SPMD in many ways.

- More strengths:
  - Well suited to emerging manycore systems.
  - Can exploit functional on-chip parallelism.
  - Can tolerate dynamic latencies.
  - Can support task/compute heterogeneity.

Task-centric/Dataflow Application Architecture Characteristics

Sandia National Laboratories

- Task execution requirements:
  - Tunable work size: Enough to efficiently use a core once scheduled.
  - Vector/SIMT capabilities.
  - Small thread-count SMP.
  - Task data dependencies.
  - Accelerator mode: Big patch.
- Universal portability:
  - Works within node, across nodes.
  - Works across heterogeneous core types.
- Many tasks:
  - Async dispatch: Many in flight.
  - Natural latency hiding.
  - Higher message injection rates.
  - Better load balancing.
- Compatible with “classics”:
  - Fortran, C, OpenMP.
  - Used within a task.
- Natural resilience model:
  - Every task has a parent (can regenerate).
- Demonstrated concept:
  - Co-Design centers, PSAAP2, others.

Open Questions for Task-Centric/Dataflow Strategies

Sandia National Laboratories

- Functional vs. Data decomposition.
  - Over-decomposition of spatial domain:
    - Clearly useful, challenging to implement.
  - Functional decomposition:
    - Easier to implement. Challenging to execute efficiently (temporal locality).
- Dependency specification mechanism.
  - How do apps specify inter-task dependencies?
  - Futures (e.g., C++, HPX), data addresses (Legion), explicit (Uintah).
- Roles & Responsibilities: App vs Libs vs Runtime vs OS.
- Interfaces between layers.
- Huge area of R&D for many years.

Data challenges:

- Read/write functions:
  - Must be task compatible.
  - Thread-safe, non-blocking, etc.
- Versioning:
  - Computation may be executing across multiple logically distinct phases (e.g. timesteps)
  - Example: Data must exist at each grid point and for all active timesteps.
- Global operations:
  - Coordination across task events.
  - Example: Completion of all writes at a time step.

Key messages:

- HPC App architectures are changing, adding further demands and opportunities for big data, big compute co-design efforts.
- Need to know HPC app trends to get combined big data, big compute right.