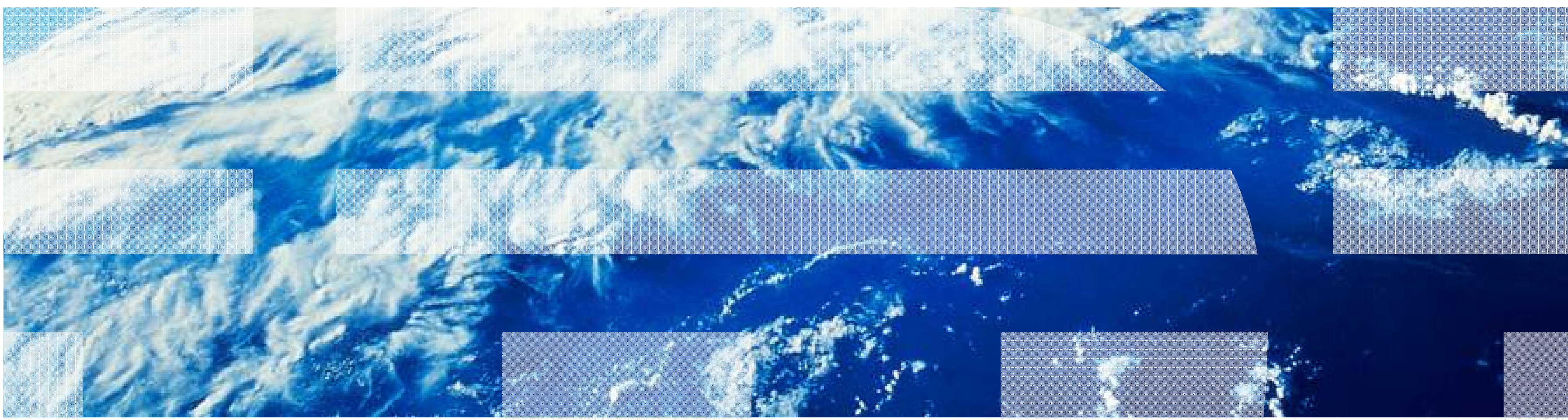
Blue Gene Active Storage for High Performance BG/Q I/O and Scalable Data-centric Analytics

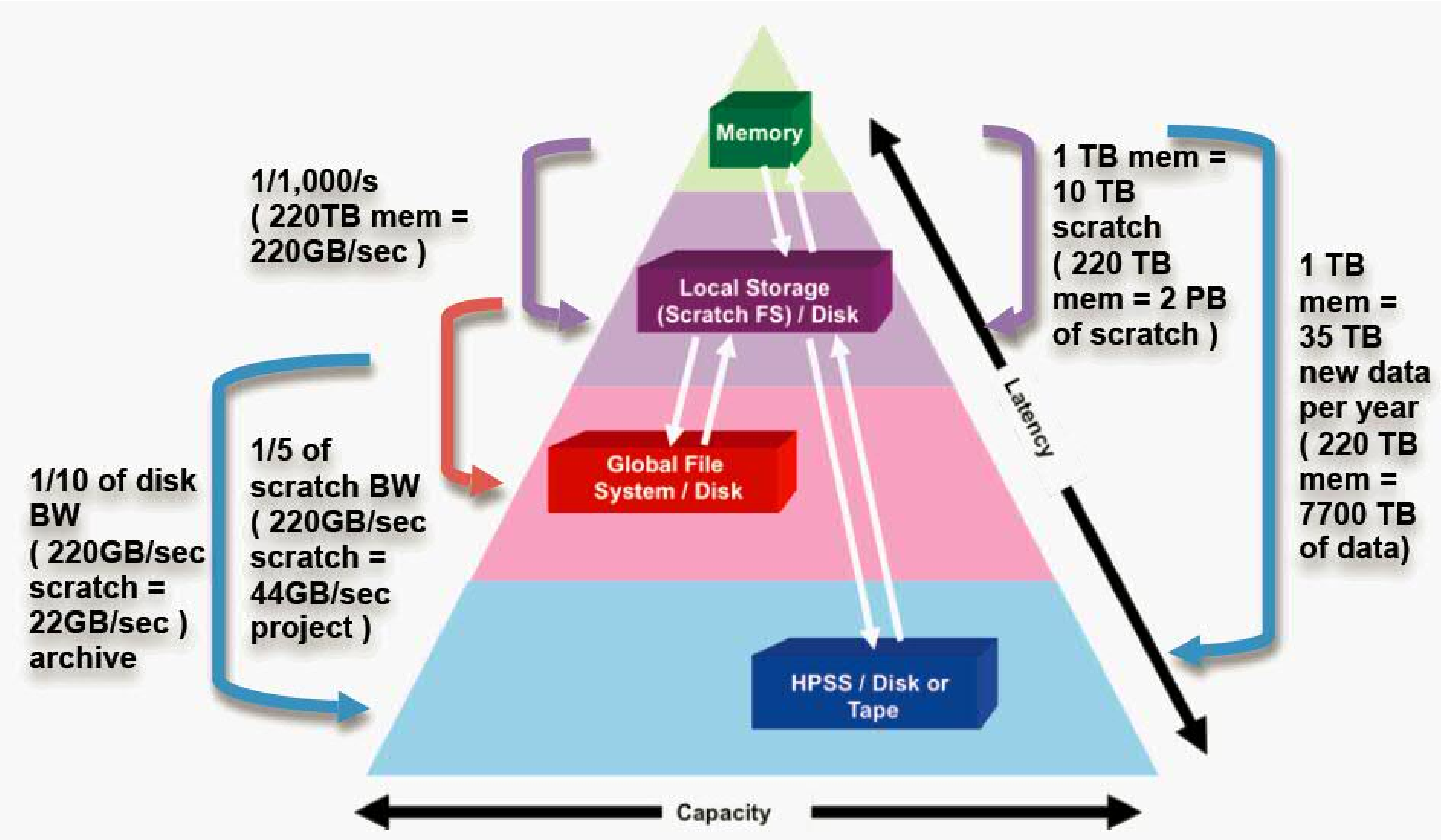
Blake G. Fitch bgf@us.ibm.com



The Extended, World Wide Active Storage Fabric Team

- Blake G. Fitch
- Robert S. Germain
- Michele Franceschini
- Todd Takken
- Bernard Metzler
- Heiko J. Schick
- Peter Morjan
- Ben Krill
- T.J. Chris Ward
- Thomas Huth
- Michael Deindl
- Michael Kaufmann
- …. and many other part time associates and contributors.

DOE Extreme Scale: Conventional Storage Planning Guidelines



http://www.nersc.gov/assets/HPC-Requirements-for-Science/HPSSExtremeScaleFINALpublic.pdf

HPC I/O Requirements – 60 TB/s – Drive

Systems	2009	2018	Difference Today & 2018
System peak	**2 Pflop/s**	**1 Eflop/s**	**O(1000)**
Power	**6 MW**	**~20 MW (goal)**	
System memory	0.3 PB	32 - 64 PB	O(100)
Node performance	125 GF	1.2 or 15TF	O(10) – O(100)
Node memory BW	25 GB/s	2 - 4TB/s	O(100)
Node concurrency	12	O(1k) or O(10k)	O(100) – O(1000)
Total Node Interconnect BW	3.5 GB/s	200-400GB/s (1:4 or 1:8 from memory BW)	O(100)
System size (nodes)	18,700	O(100,000) or O(1M)	O(10) – O(100)
Total concurrency	225,000	O(billion) + [O(10) to O(100) for latency hiding]	O(10,000)
Storage Capacity	15 PB	500-1000 PB (>10x system memory is min)	O(10) – O(100)
IO Rates	0.2 TB	$\boxed{\text{60 TB/s}}$	O(100)
MTTI	days	O(1 day)	- O(10)

From Rick Stevens: http://www.exascale.org/mediawiki/images/d/db/PlanningForExascaleApps-Steven.pdf

Exascale IO: 60TB/s Drives Use of Non-volatile Memory

- 60 TB/s bandwidth required
  - Driven by higher frequency check points due to low MTTI
  - Driven by tier 1 file system requirements
    - HPC program IO
    - Scientific analytics at exascale

- Disks:
  - ~100 MB/s per disk
  - ~600,000 disks!
  - ~600 racks???

- Flash
  - ~100 MBps write bandwidth per flash package
  - ~600,000 Flash packages
  - ~60 Flash packages per device
  - ~6 GBps bandwidth per device (e.g. PCIe 3.0 x8 Flash adaptor)
  - ~10,000 Flash devices

- Flash is already more cost effective than disk for performance (if not capacity) at the unit level and this effect is amplified by deployment requirements (power, packaging, cooling)

Conclusion: Exascale class systems will benefit from integration of a large solid state storage subsystem

An Increasing Number Scalable Programming Models

Random Comms

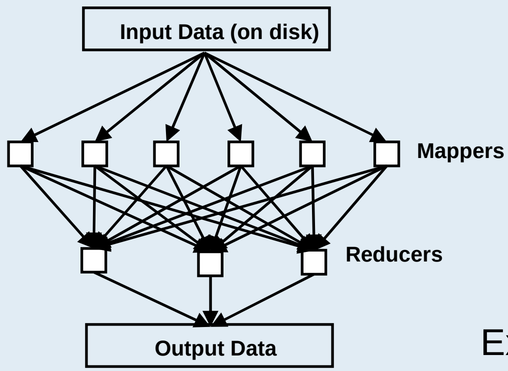
Structured Comms

Embarrassingly Parallel

Network Dependent

Data Intensive: Data at Rest

JAQL, Java



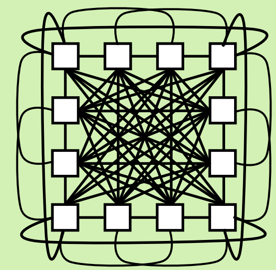
Data at Rest*:
- **High Volume**
- **Mixed Variety**
- **Low Velocity**

Extreme Scale-out

(*pre-partitioned)

Data and Compute Intensive (Large Address Space)

C/C++, Fortran, UPC, SHMEM, MPI, OpenMP



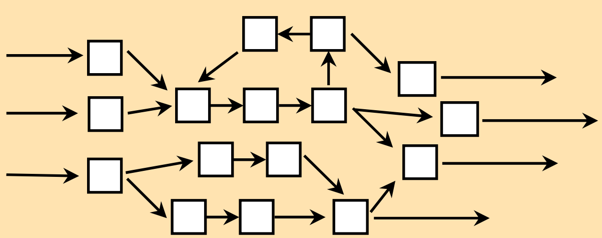
Data is Moving:
- **Long Running**
- **All Data View**
- **Small Messages**

Discrete Math
Low Spatial Locality

Data Intensive: Streaming Data

SPL, C, Java

Data in Motion:
- **High Velocity**
- **Mixed Variety**
- **High Volume***

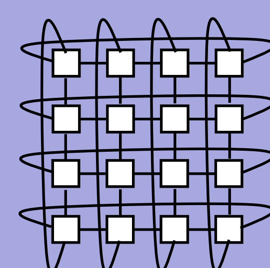


Reactive Analytics
Extreme Ingestion

(*over time)

Compute Intensive (Data Generators)

C/C++, Fortran, MPI, OpenMP



Data is Generated:
- **Long Running**
- **Small Input**
- **Massive Output**

Generative Modeling
Extreme Physics

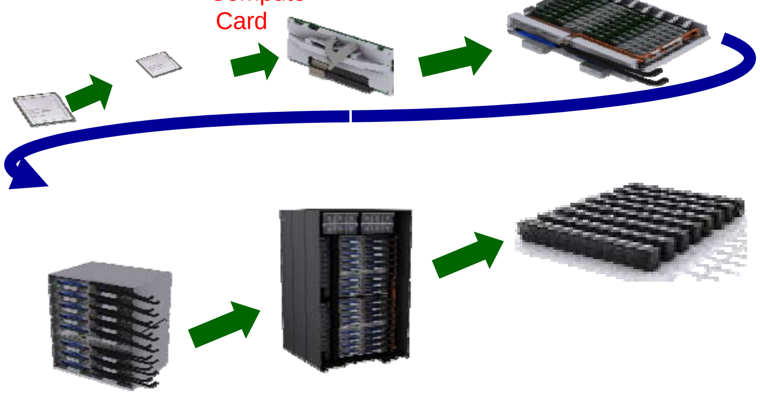
= compute node

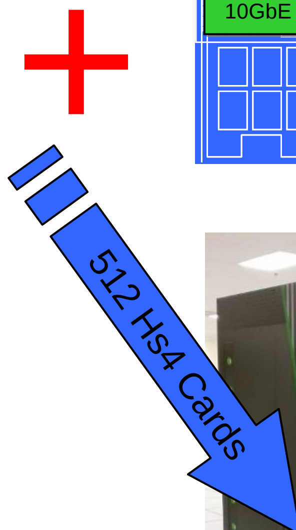
Active Storage Concept: Scalable, Solid State Storage with BG/Q

"How to" guide:

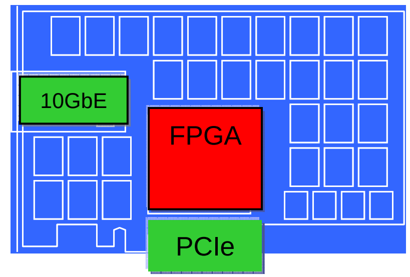
- Remove 512 of 1024 BG/Q compute nodes in rack – to make room for solid state storage
- Integrate 512 Solid State (Flash+) Storage Cards in BG/Q compute node form factor

Standard BG/Q Compute Fabric





PCIe Flash Board



Flash Storage	2012 Targets
Capacity	2 TB
I/O Bandwidth	2 GB/s
IOPS	200 K

BGAS Rack Targets

Nodes	512
Storage Cap	1 PB
I/O Bandwidth	1 TB/s
Random IOPS	100 Million
Compute Power	104 TF
Network Bisect.	512 GB/s
External 10GbE	512

… scale it like BG/Q.

System Software Environment

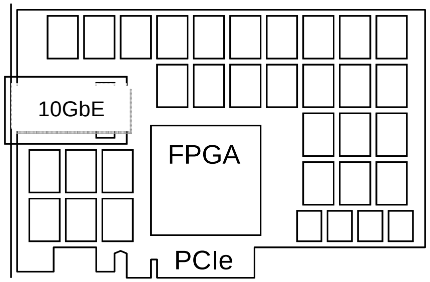
- Linux OS enabling storage + embedded compute
- OFED RDMA & TCP/IP over BG/Q Torus – failure resilient
- Standard middleware – GPFS, DB2, MapReduce, Streams

Active Storage Target Applications

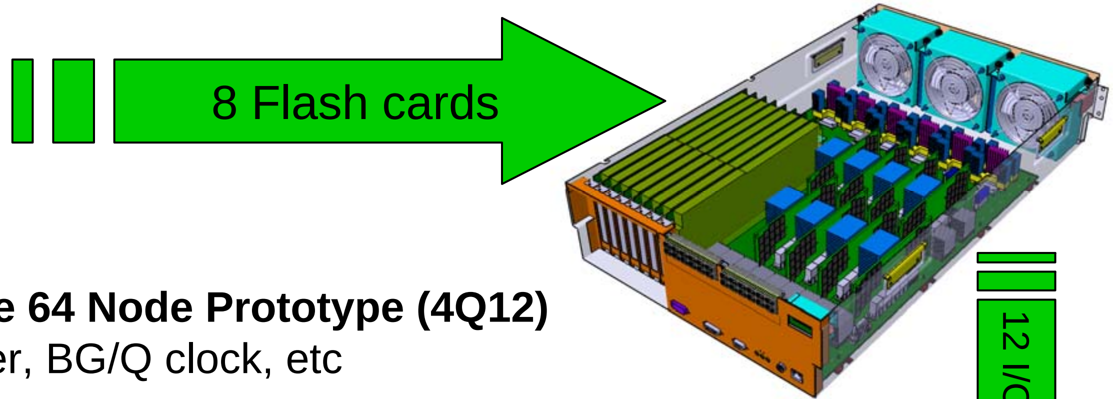
- Parallel File and Object Storage Systems
- Graph, Join, Sort, order-by, group-by, MR, aggregation
- Application specific storage interface

Key architectural balance point: All-to-all throughput roughly equivalent to Flash throughput

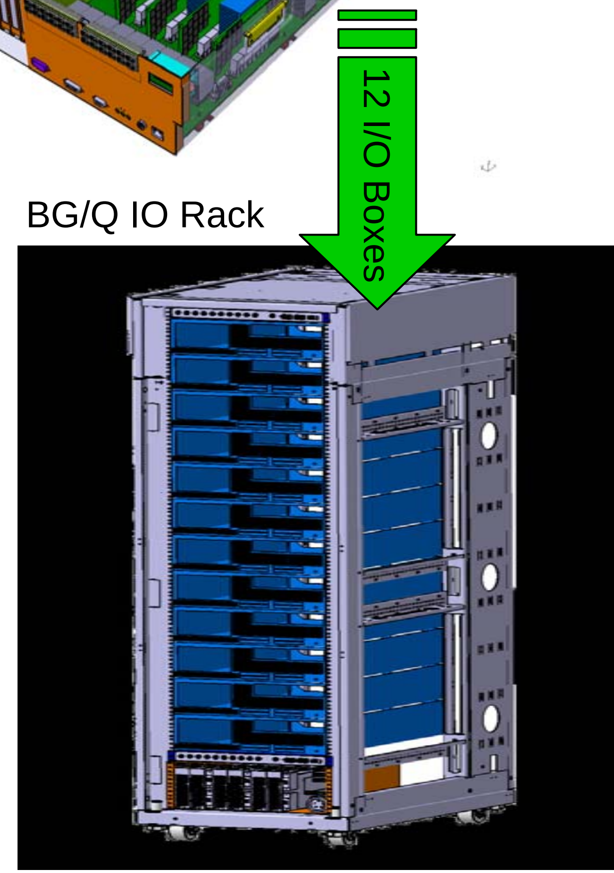
Scalable PCIe Flash device



BG/Q I/O Drawer (8 nodes)



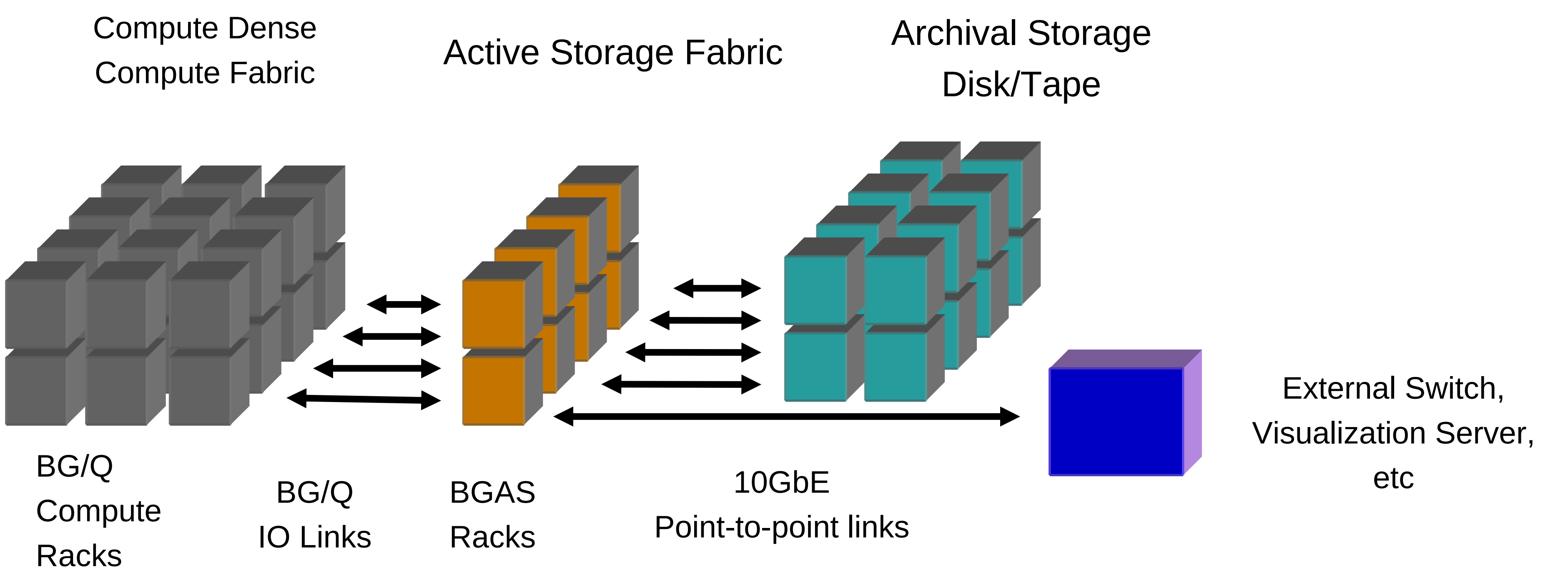
BG/Q IO Rack



Blue Gene Active Storage 64 Node Prototype (4Q12)

- IBM 19 inch rack w/ power, BG/Q clock, etc
- PowerPC Service Node
- 8 IO Drawers
- 3D Torus (4x4x4)
- System specification targets:
  - 64 BG/Q Nodes – 12.8TF
  - 128 TB Flash Capacity (SLC)
  - 128 GB/s Flash I/O Bandwidth
  - 128 GB/s Network Bisection Bandwidth
  - 4 GB/s Per node All-to-all Capability
  - 128x10GbE External Connectivity
  - 256 GB/s I/O Link Bandwidth to BG/Q Compute Nodes
- Software Targets
  - Linux 2.6.32
  - TCP/IP, OFED RDMA
  - GPFS
  - MVAPICH
  - SLURM

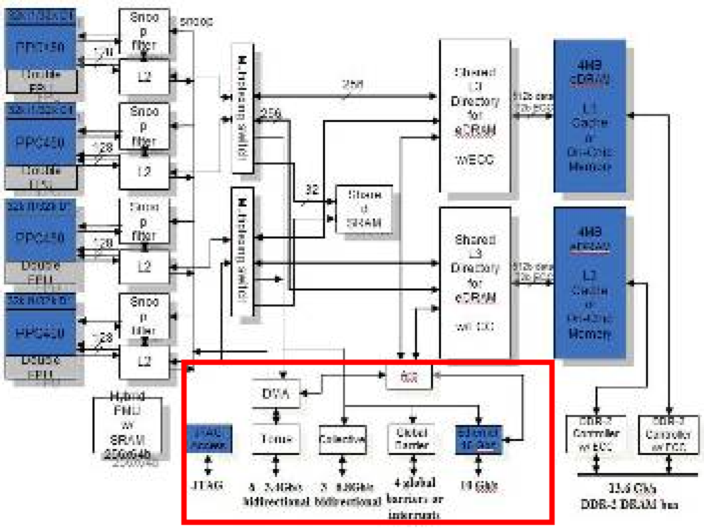
Exascale System Architecture Prototype by Extending BG/Q



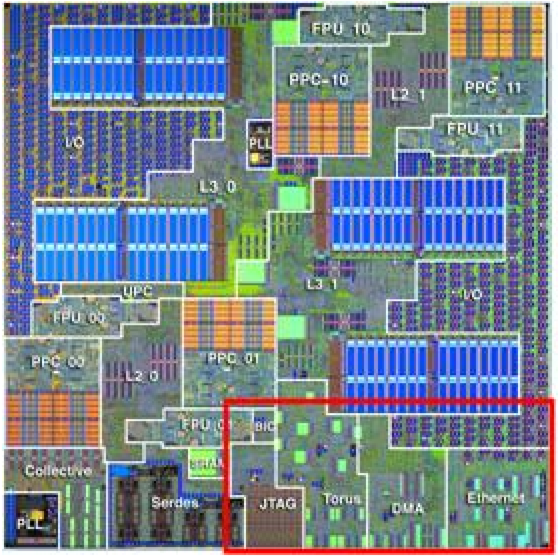
- IO nodes have non-volatile memory as storage and external Ethernet
- Compute nodes not required for data-centric systems but offer higher density for HPC
- Compute nodes and IO nodes likely have Blue Gene type nodes and torus network
- IO Node cluster supports “file systems” in non-volatile memory and on disk
- On-line data does may not leave IO fabric until ready for long term archive
- Manual or automated hierarchical storage manages data object migration among tiers

Torus Networks – Cost-scalable To Thousands Of Nodes

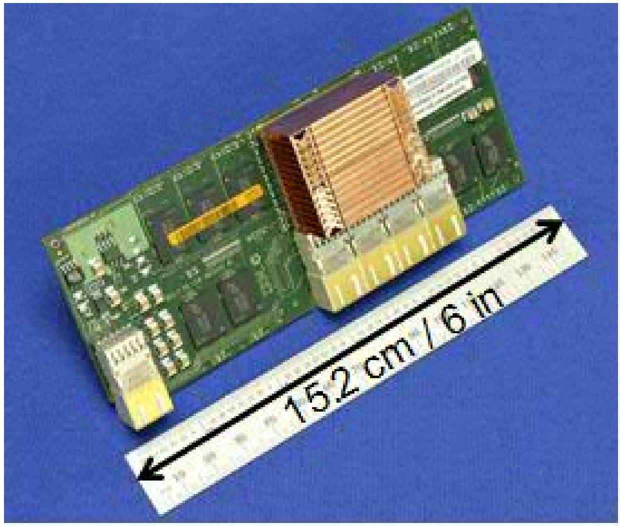
Logic Diagram



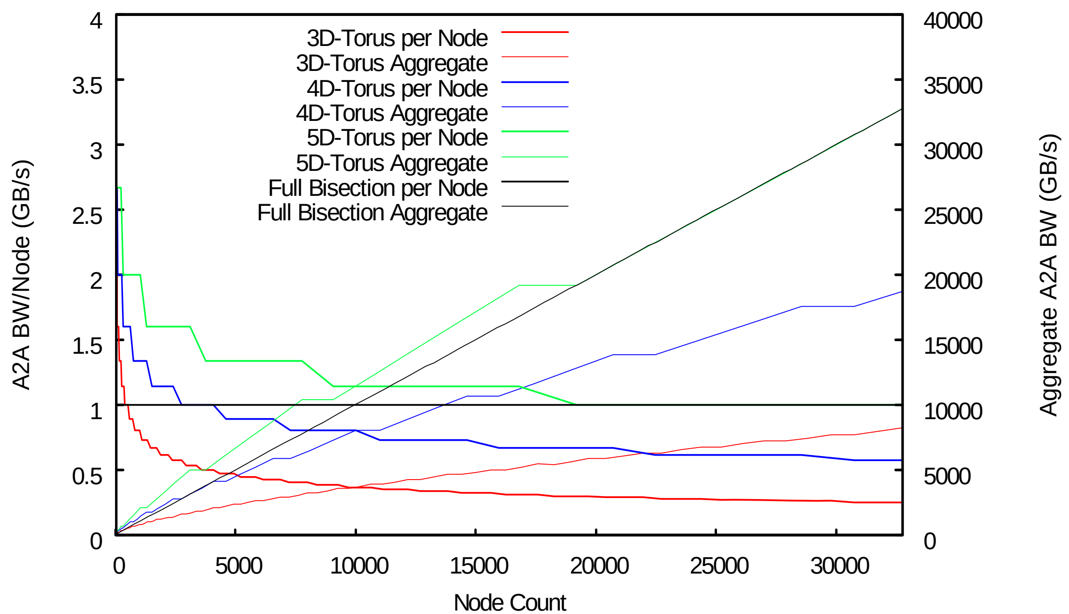
Physical Layout



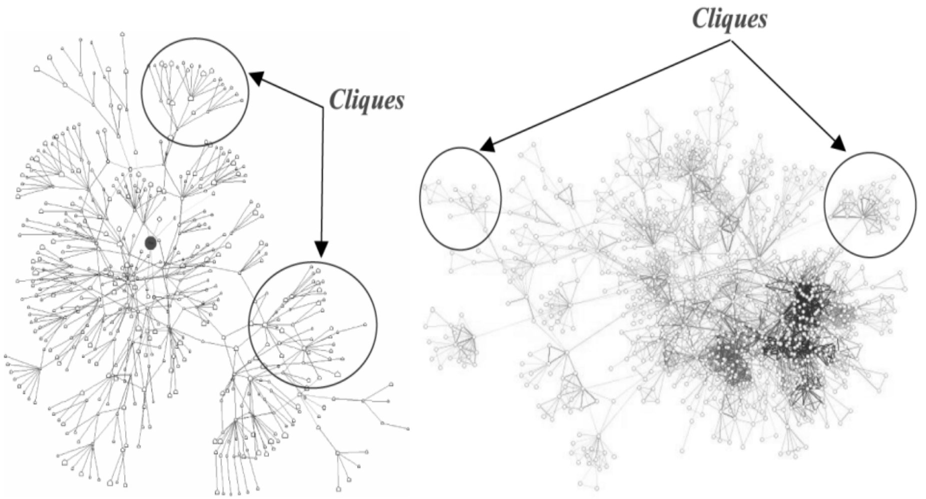
Packaged Node



- Cost scales linearly with number of nodes
- Torus all-to-all throughput does fall rapidly for very small system sizes
- But, bisectional bandwidth continues to rise as system grows
- A hypothetical 5D Torus with 1GB/s links yields theoretical peak all-to-all bandwidth of:
  - 1GB/s (1 link) at 32k nodes (8x8x8x8x8)
  - Above 0.5GB/s out to 1M nodes
- **Mesh/Torus networks can be effective for data intensive applications where cost/bisection-bw is required**



Graph 500: Demonstrating Blue Gene/Q Data-centric Capability



(a) Graph of a Local Gnutella P2P **(b)** Graph of Research Co-authorship

- **A particularly important analytics kernel**
  - Random memory access pattern
    - Very fine access granularity
  - High load imbalance in
    - Communication and
    - Computation
  - Data dependent communications patterns

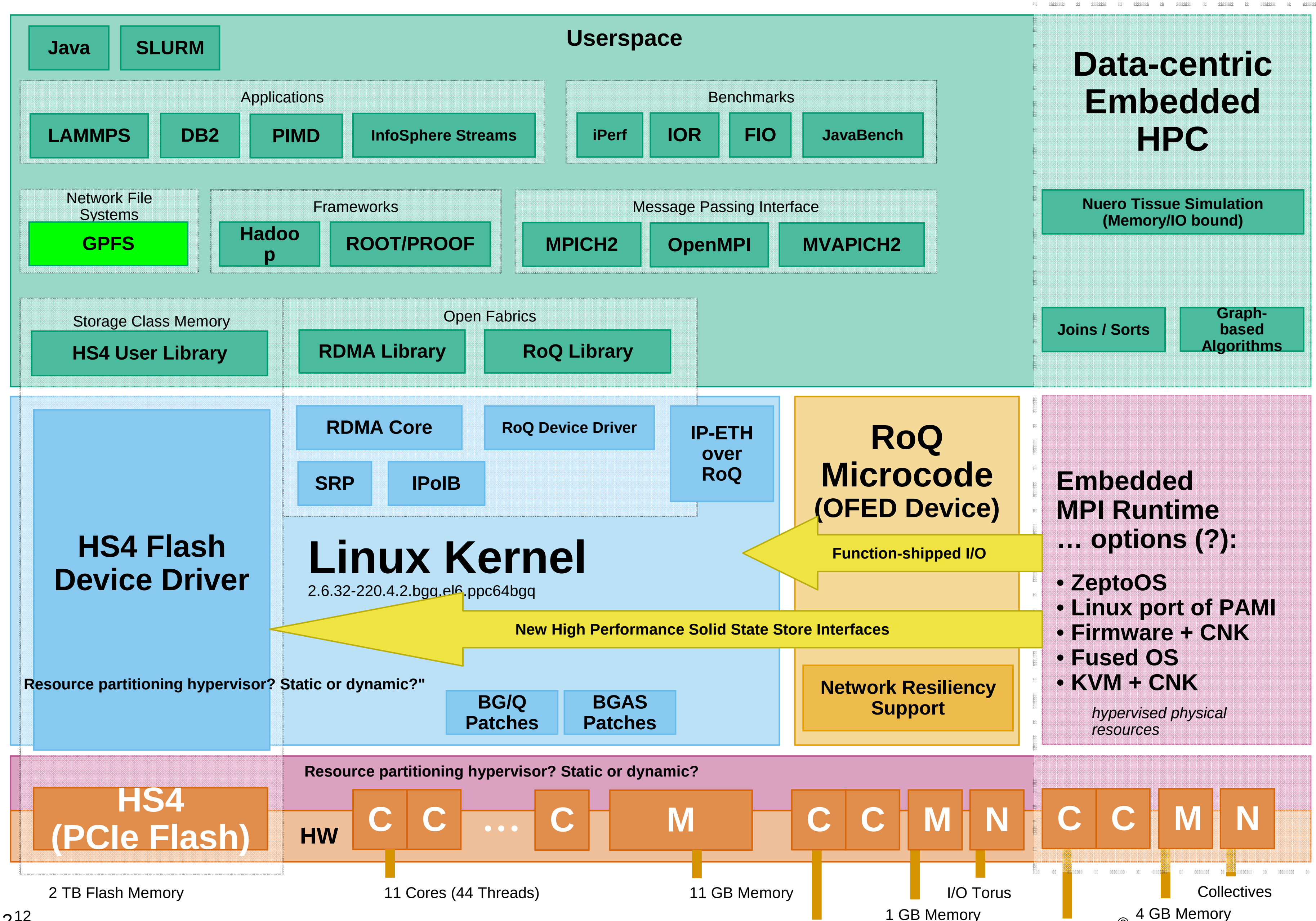
- **Blue Gene features which helped:**
  - cost-scalable bisectional bandwidth
  - low latency network with high messaging rates
  - large system memory capacity
  - low latency memory systems on individual nodes

June '12: www.graph500.org/results_june_2012

June 2012

Rank	Installation Site	Machine	Number of nodes	Number of cores	Problem scale	GTEPS
1	DOE/SC/Argonne National Laboratory	Mira/BlueGene/Q	32768	524288	38	3541.00
1	LLNL	Sequoia/Blue Gene/Q	32768	524288	38	3541.00
2	DARPA Trial Subset, IBM Development Engineering	Power 775, POWER7 8C 3.836 GHz	1024	32768	35	508.05
3	Information Technology Center, The University of Tokyo	Oakleaf-FX (Fujitsu PRIMEHPC FX 10)	4800	76800	38	358.10
4	GSIC Center, Tokyo Institute of Technology	HP Cluster Platform SL390s G7 (three Tesla cards per node)	1366	16392	35	317.09
5	Brookhaven National Laboratory	BLUE GENE/Q	1024	16384	34	294.29
6	DOE/SC/Argonne National Laboratory	Vesta/BlueGene/Q	1024	16384	34	292.36
7	NASA-Ames / Parallel Computing Lab, Intel Labs	Pleiades - SGI ICE-X, dual plane hypercube FDR infiniband, E5-2670 "sandybridge"	1024	16384	34	270.33
8	NERSC/LBNL	XE6	4817	115600	35	254.07
9	NNSA and IBM Research, T.J. Watson	NNSA/SC Blue Gene/Q Prototype II	4096	65536	32	236.00
10	GSIC Center, Tokyo Institute of Technology	TSUBAME 2.0 (CPU only)	1366	16392	36	202.68

RedHat Linux Based Active Storage Software Stack



BGAS Full System Emulator -- BGAS on BG/Q Compute Nodes

- Leverages BG/Q Active Storage (BGAS) Environment
  - BG/Q + Flash memory,
  - Linux REHL 6.2
  - standard network interfaces (OFED RDMA, TCP/IP)
  - standard middleware (GPFS, etc)

- BGAS environment + Soft Flash Controller + Flash Emulator
  - SFC breaks the work up between device driver and FPGA logic
  - The Flash emulator manages RAM as storage with Flash access times

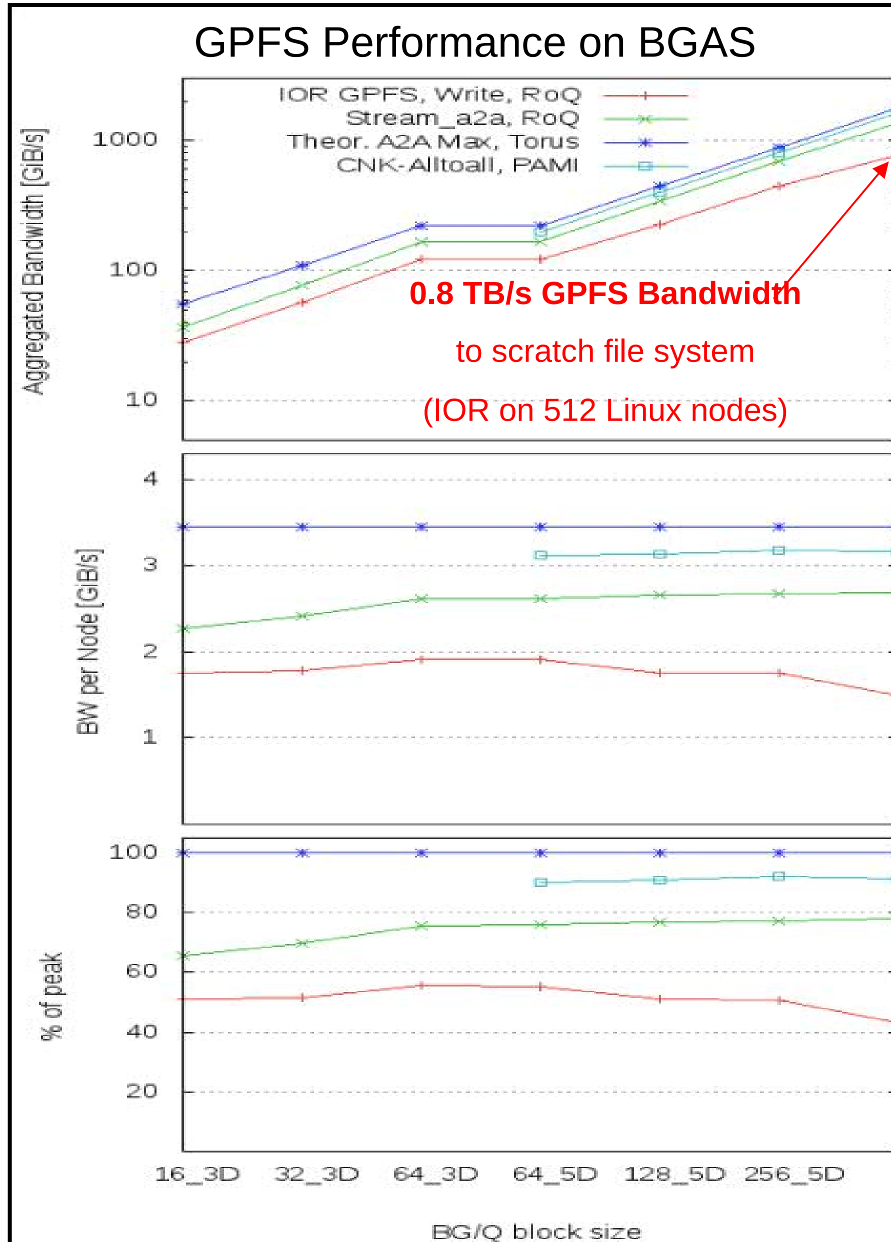
- Explore scalable, SoC, SCM (Flash, PCM) challenges and opportunities
  - Work with the many device queues necessary for BG/Q performance
  - Work on the software interface between network and SCM
    - RDMA direct into SCM

- Realistic BGAS development platform
  - Allows development of storage systems that deal with BGAS challenges
    - GPFS-SNC should run
    - GPFS-Perseus type declustered raid
  - Multi-job platform challenges
    - QoS requirements on the network
    - Resiliency in network, SCM, and cluster system software

- Running today: InfoSphere Streams, DB2, GPFS, Hadoop, MPI, etc…

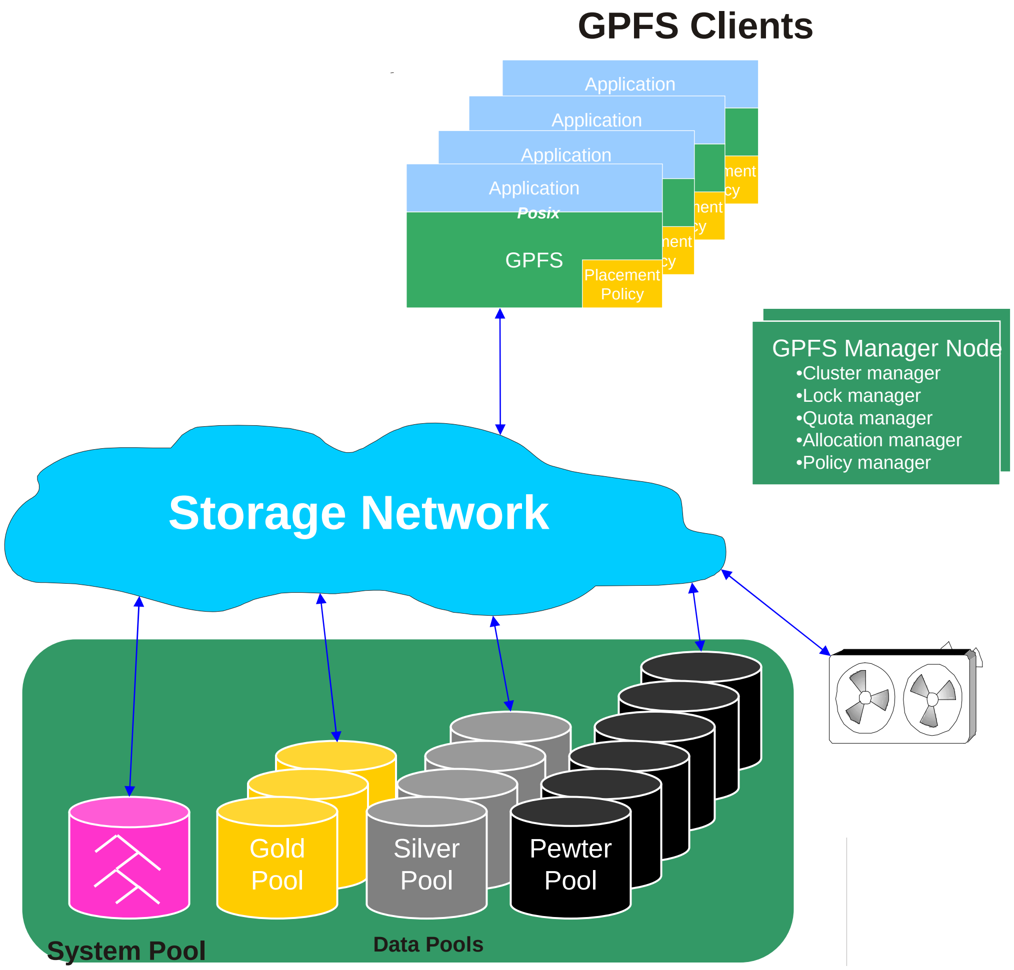
BGAS Platform Performance – Emulated Storage Class Memory

- Software Environment
  - Linux Operating System
  - OFED RDMA on BG/Q Torus Network
  - Fraction of 16 GB DRAM used to emulate Flash storage (RamDisk) on each node
  - GPFS uses emulated Flash to create a global shared file system
- Tests
  - IOR Standard Benchmark
    - all nodes do large contiguous writes – tests A2A BW internal to GPFS
  - All-to-all OFED RDMA verbs interface
  - MPI All-to-all in BG/Q product environment – a light weight compute node kernel (CNK)
- Results
  - IOR used to benchmark GPFS
    - 512 nodes → 0.8 TB/s bandwidth to emulated storage
- Network software efficiency for all-to-all
  - BG/Q MPI on CNK: 95%
  - OFED RDMA verbs 80%
  - GPFS IOR 40% - 50% (room to improve!)



GPFS Information Lifecycle Management (ILM)

- **GPFS ILM abstractions:**
  - Storage pool – a group of storage volumes (disk or tape)
  - Policy – rules for placing files into storage pools
- **GPFS policy rules much richer than conventional HSM "how big is the file and when was it last touched"**
  - Tiered storage – create files on fast, reliable storage (e.g. solid state), move files as they age to slower storage, then to tape (a.k.a. HSM)
  - Differentiated storage - place media files on storage with high throughput, database on storage with high IO's per second
  - Grouping - keep related files together, e.g. for failure containment or project storage



Blue Gene Active Storage Use Models

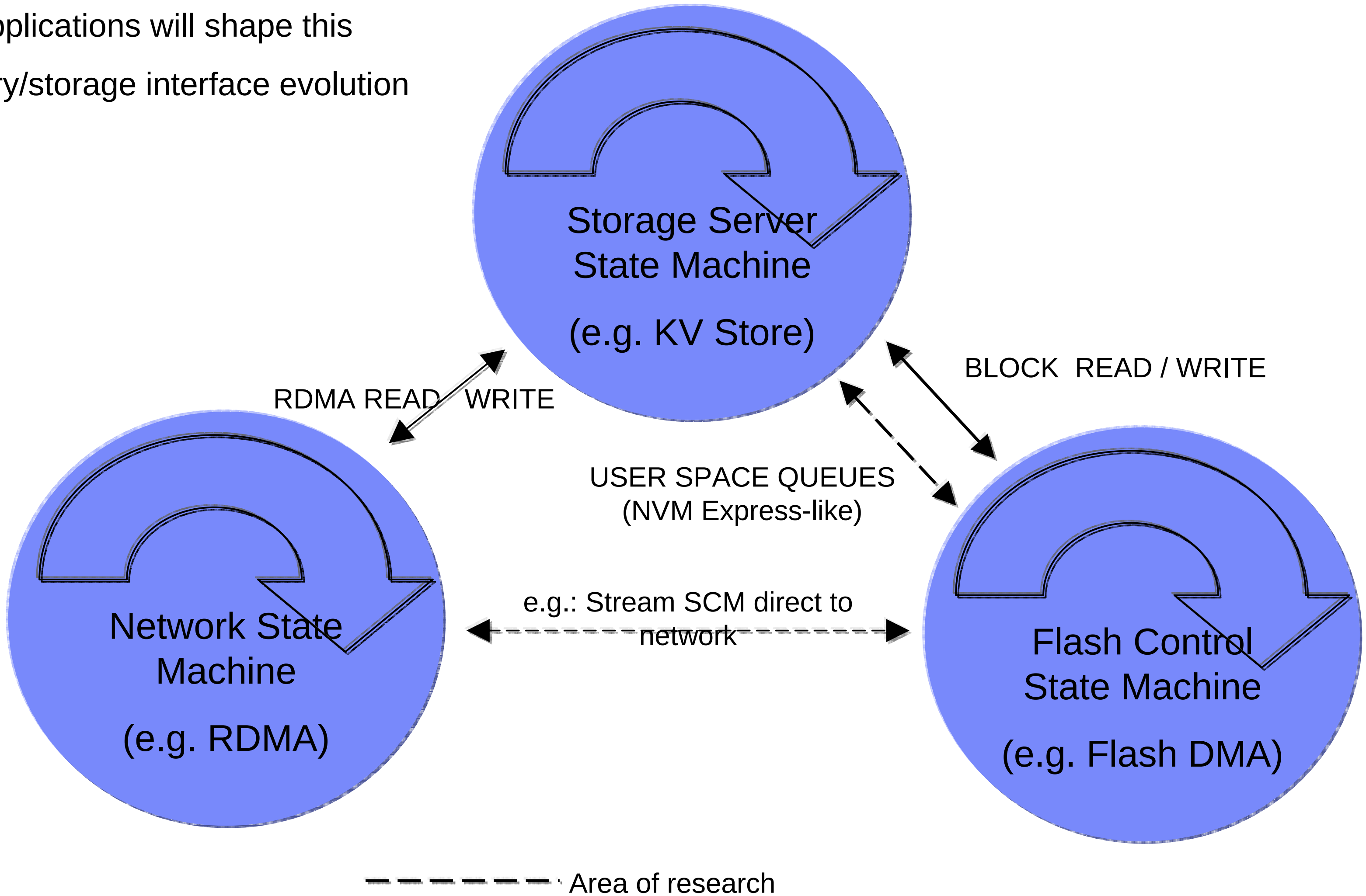
- Data Intensive Supercomputing (HPC)
  - Integrate with standard BG/Q system as standard Posix I/O accelerators
  - Create/modify HPC applications to make direct use of new capabilities ex: neuro-simulation
  - Low latency storage access from compute dense
  - New opportunities for out-of-core programming techniques

- Standard Middleware
  - BGAS utilized as a standard cluster with very high performance
  - Configure standard middleware such as GPFS, DB2, etc to run in BGAS environment

- New Frameworks
  - Restructured HPC applications and workflows to use new middleware to intercommunicate
  - Acceleration
    - Active File System to offload UNIX commands into BGAS
    - DB2 offload via Infosphere Federated Wrapper to offload and accelerate relational operators
  - InfoSphere Streams

Active Storage Stack Optimizes Network/Storage/NVM Data Path

- Scalable, active storage currently involves three server side state machines
  - Network (TCP/IP, OFED RDMA), Storage Server (GPFS, PIMD, etc), and Solid State Store (Flash Cntl)
- These state machines will evolve and potentially merge as to better server scalable, data-intensive applications.
- Early applications will shape this memory/storage interface evolution



Parallel In-Memory Database (PIMD)

- Key/value object store
  - Similar in function to Berkeley DB
  - Support for “partitioned datasets” – named containers for groups of K/V records
  - Variable length key, variable length value
  - Record values maybe appended to, or accessed/updated by byte range
  - Data consistency enforced at the record level by default

- In-Memory – DRAM and/or Non-volatile memory (with migration to disk supported).

- Other functions
  - Several types of interators from generic next-record to “streaming, parallel, sorted” keys
  - Sub-record projections
  - Bulk insert
  - Server controlled embedded function -- could include further push down into FPGA

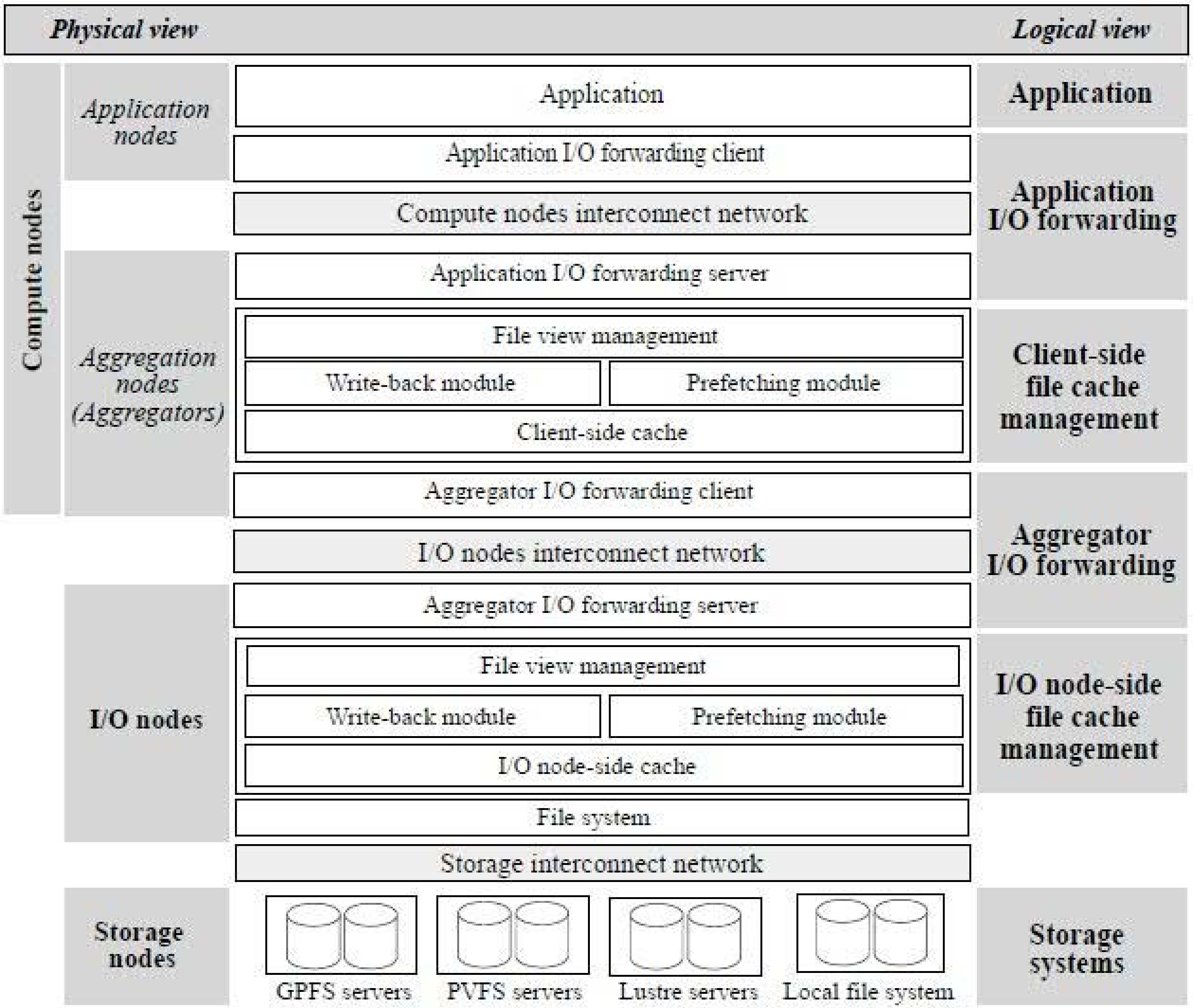
- Parallel Client/Server Storage System
  - Server is a state machine is driven by OFED RDMA and Storage events
  - MPI client library wraps OFED RDMA connections to servers

- Hashed data distribution
  - Generally private hash to avoid data imbalance in servers
  - Considering allowing user data placement with a maximum allocation at each server

- Resiliency
  - Currently used for scratch storage which is serialized into files for resiliency
  - Plan to enable scratch, replication, and network raid resiliency on PDS granularity

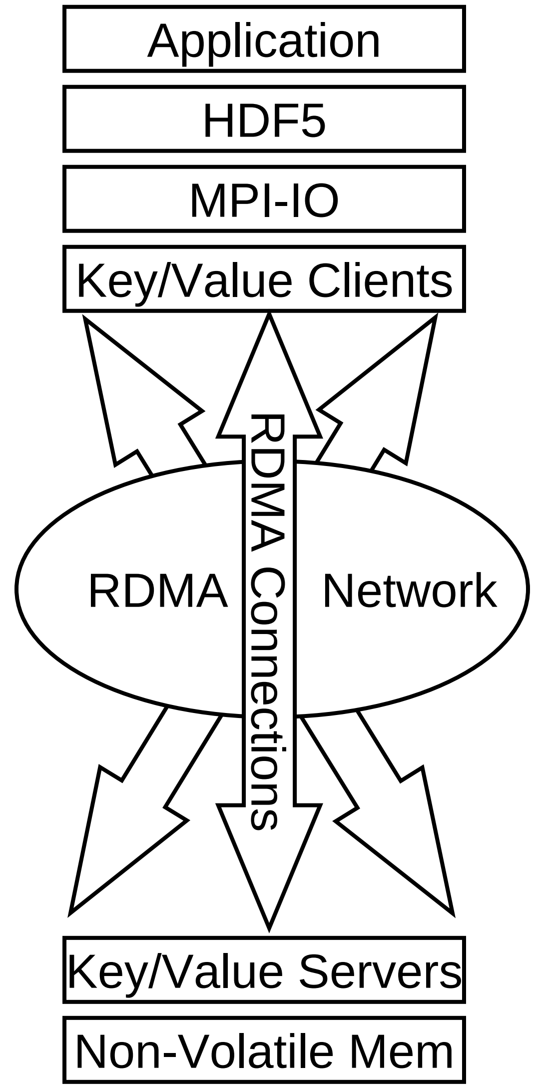
The Classic Parallel I/O Stack v. a Compute-In-Storage Approach

Classic parallel IO stack to access external storage



From: http://e-archivo.uc3m.es/bitstream/10016/9190/1/thesis_fjblas.pdf

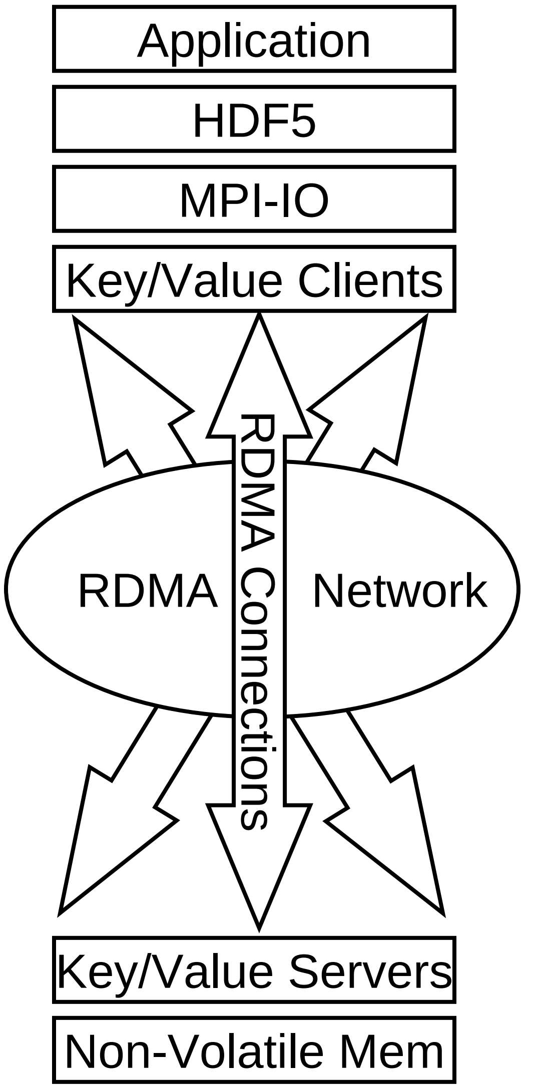
Compute-in-storage Apps directly connect to scalable K/V storage



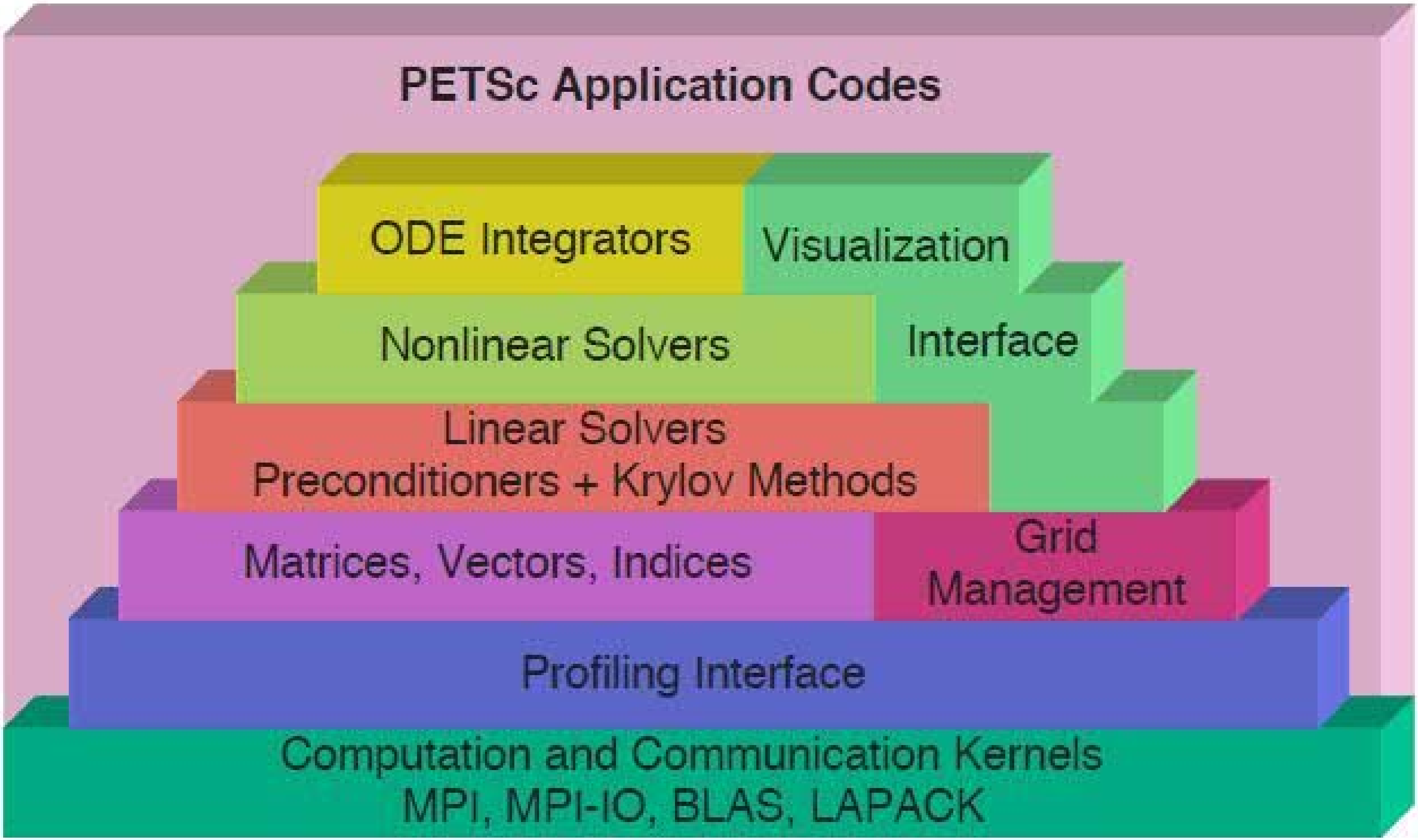
Compute-in-Storage: HDF5 Mapped to a Scalable Key/Value Inteface

- Storage-embedded parallel programs can use HDF5
  – Many scientific packages already use HDF5 for I/O
- HDF5 mapped scalable key/value storage (SKV)
  – client interface:
    - native key/value
    - tuple representation of records
    - MPI/IO (adio)
  – --> support for high-level APIs: HDF5
  – --> broad range of applications
- SKV provides lightweight direct access to NVM
- client-server communications use OFED RDMA
- Scalable Key/Value storage (SKV)
  – Design
    - stores key/value records in (non-volatile) memory
    - distributed parallel client-server
    - thin server core: mediate between network and storage
    - client access: RDMA only
  – Features:
    - non-blocking requests (deep queues)
    - global/local iterators
    - access to partial values (insert/retrieve/update/append)
    - tuple-based access with server side predicates and projection

Compute-in-storage Apps directly connect to scalable K/V storage



Structure Of PETSc

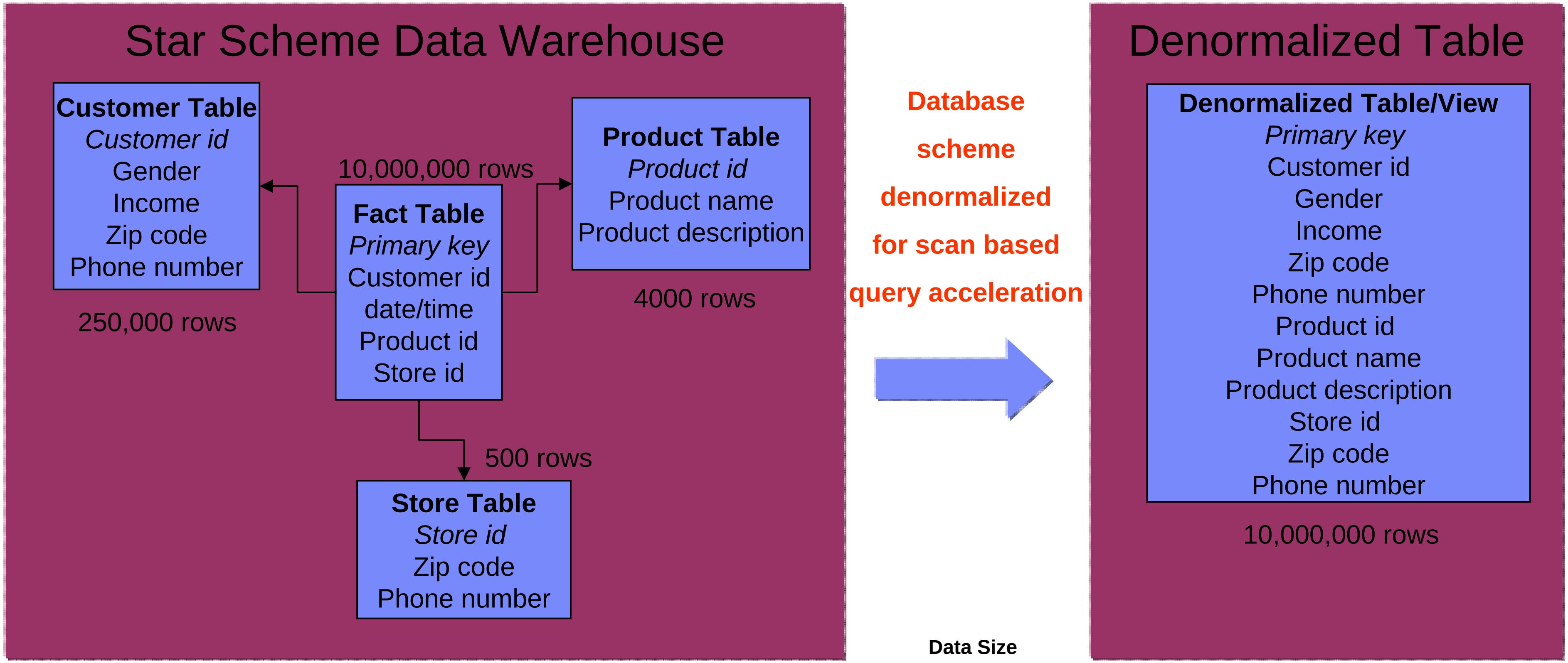


From: http://www.it.uu.se/research/conf/SCSE07/material/Gropp.pdf

Query Acceleration: Parallel Join vs Distributed Scan

Total size in normal form:
4×10^8 Bytes
(Most compact representation)

Total size in denormalized form:
1.2×10^{10} Bytes
(60X growth in data volume)

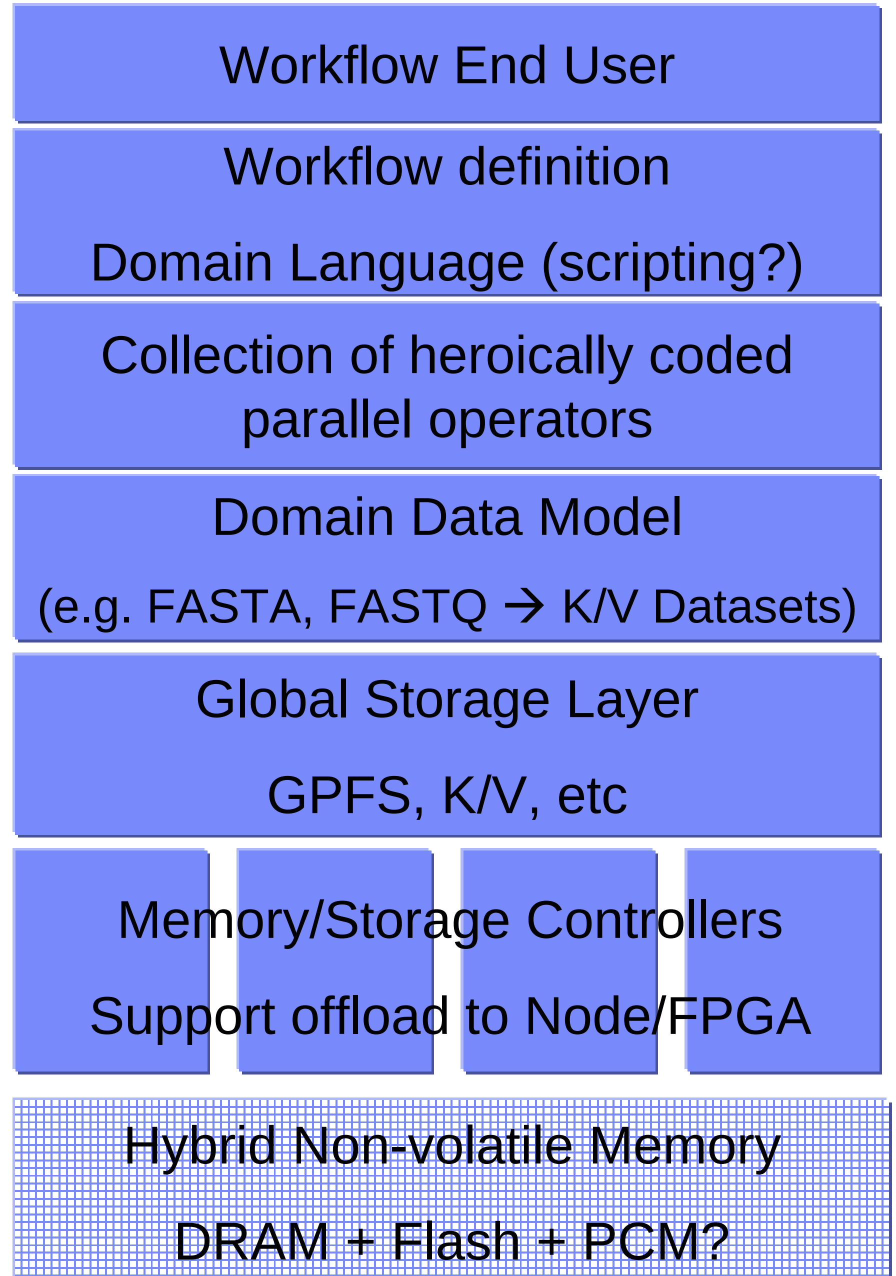


- Scheme denormalization is often done for query acceleration in systems with weak or nonexistant network bisection.
- Strong network bisection enables efficient, ad-hoc table joins allowing data to remain in compact normla form.

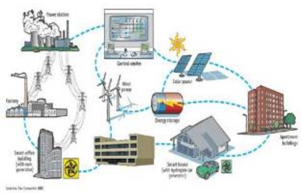
Data Size
Primary key – 8 bytes
Customer id – 8 bytes
Date/time – 11 bytes
Product id – 8 bytes
Store id – 4 bytes
Gender – 1 byte
Zip code – 9 bytes
Phone number – 10 bytes
Product name – varchar(128)
Product description – varchar(1024)

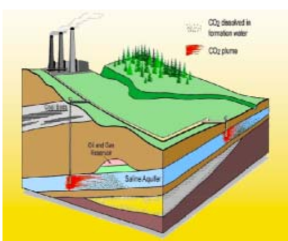
Observations, comments, and questions

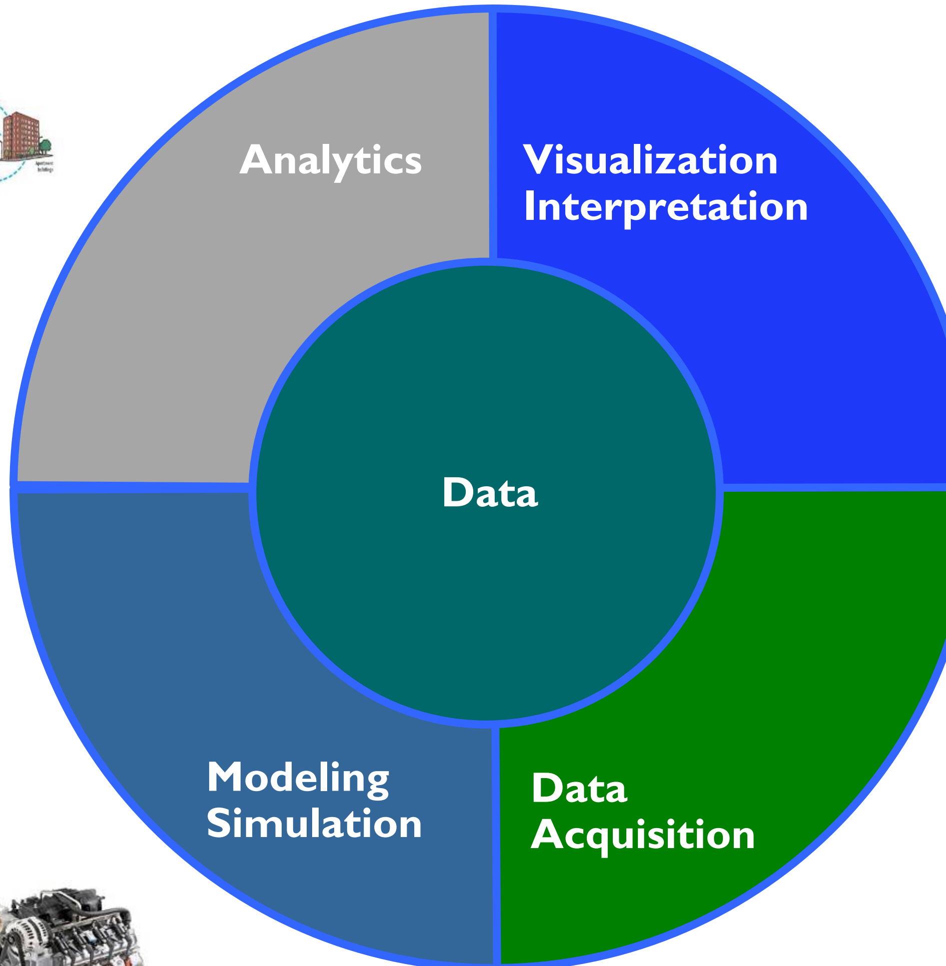
- How big are the future datasets? How random are the accesses? How much concurrency in algorithms?
- Heroic programming will be probably be required to make 100,000 node programs work well – what about down scaling?
- A program using a library will usually call multiple interfaces during its execution life cycle – what are the options for data distribution?
- Domain workflow design may not be the same skill as building a scalable parallel operator – what will it will take care to enable these activities to be independent?
- A program using a library will only spend part of its execution time in the library – can/must parallel operators in the library be pipelined or execute in parallel?
- Load balance – who's responsibility?
- Are interactive workflows needed? Would it help to reschedule a workflow while a person thinks to avoid speculative runs?
- Operator pushdown – how far?
  - There is higher bandwidth and lower latency in the parallel storage array than outside, in the node than on the network, in the storage controller (FPGA) than in the node
  - Push operators close to data but keep a good network around for when that isn't possible



End to End Analysis







Massive Scale Data and Compute