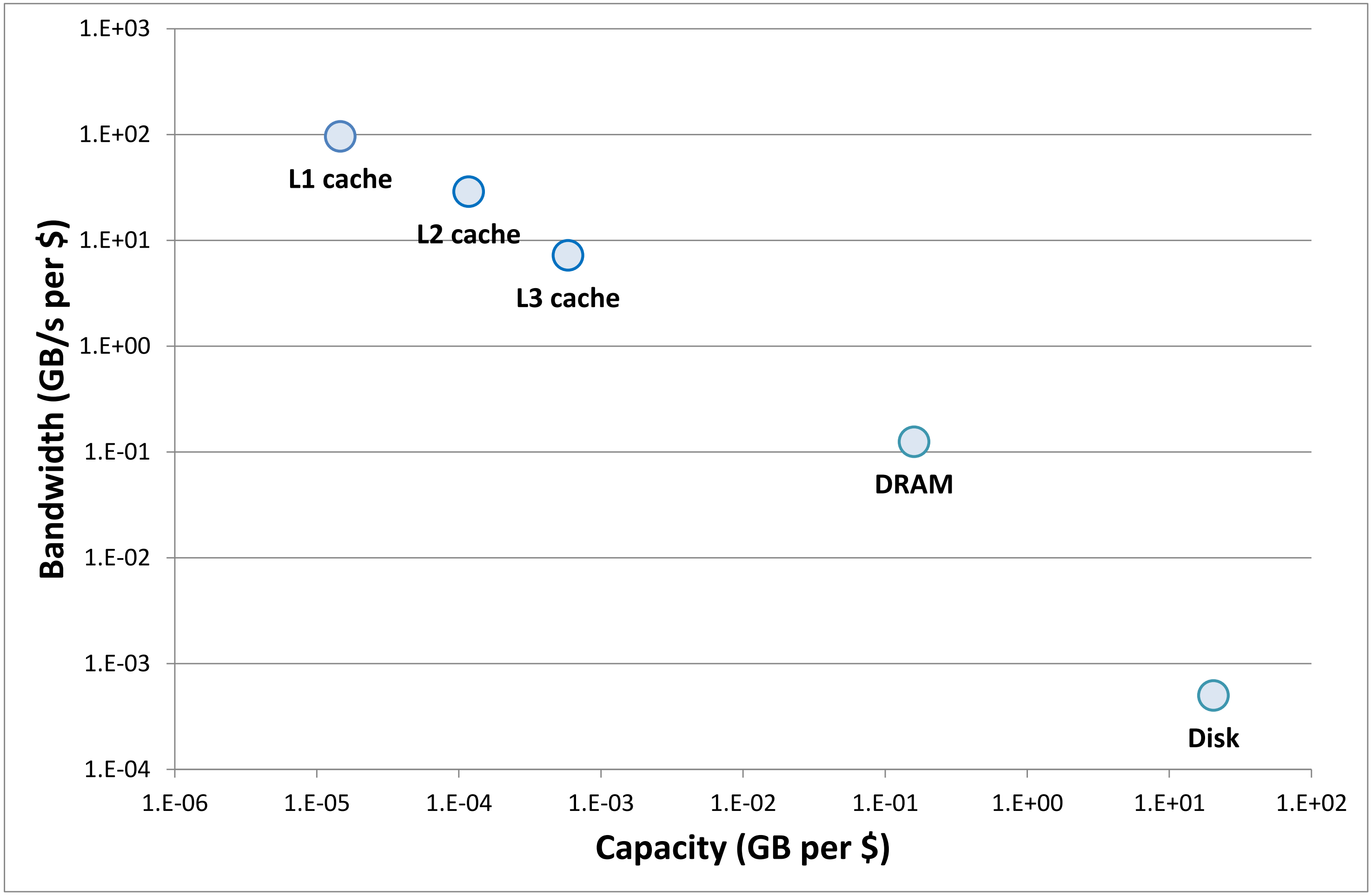
Programming an Extended Memory Hierarchy

Duncan Roweth
Cray CTO Office

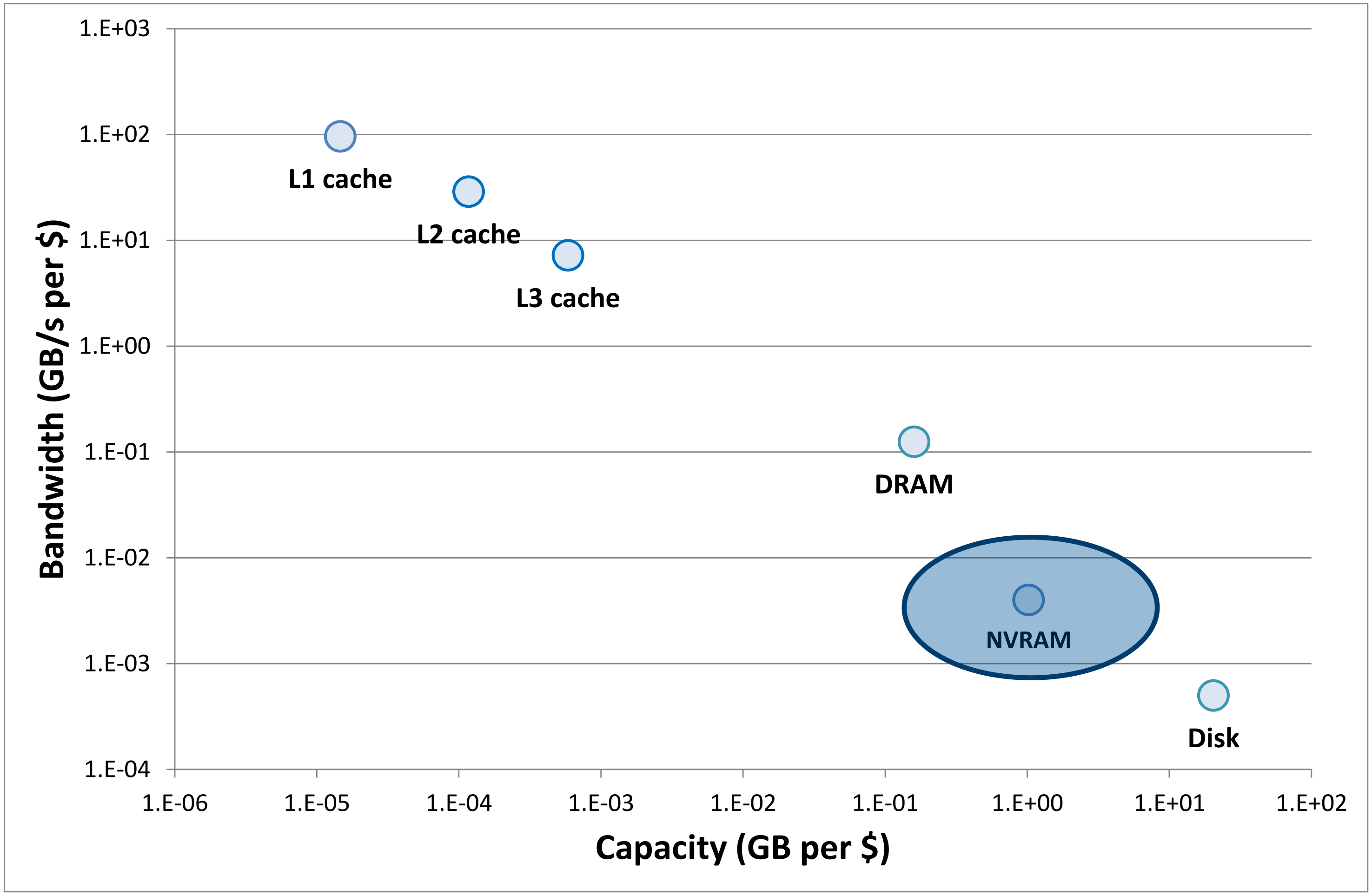
Outline of my talk

- New elements of memory hierarchy
- Use cases
- Ideas on programming model integration
- Conclusions

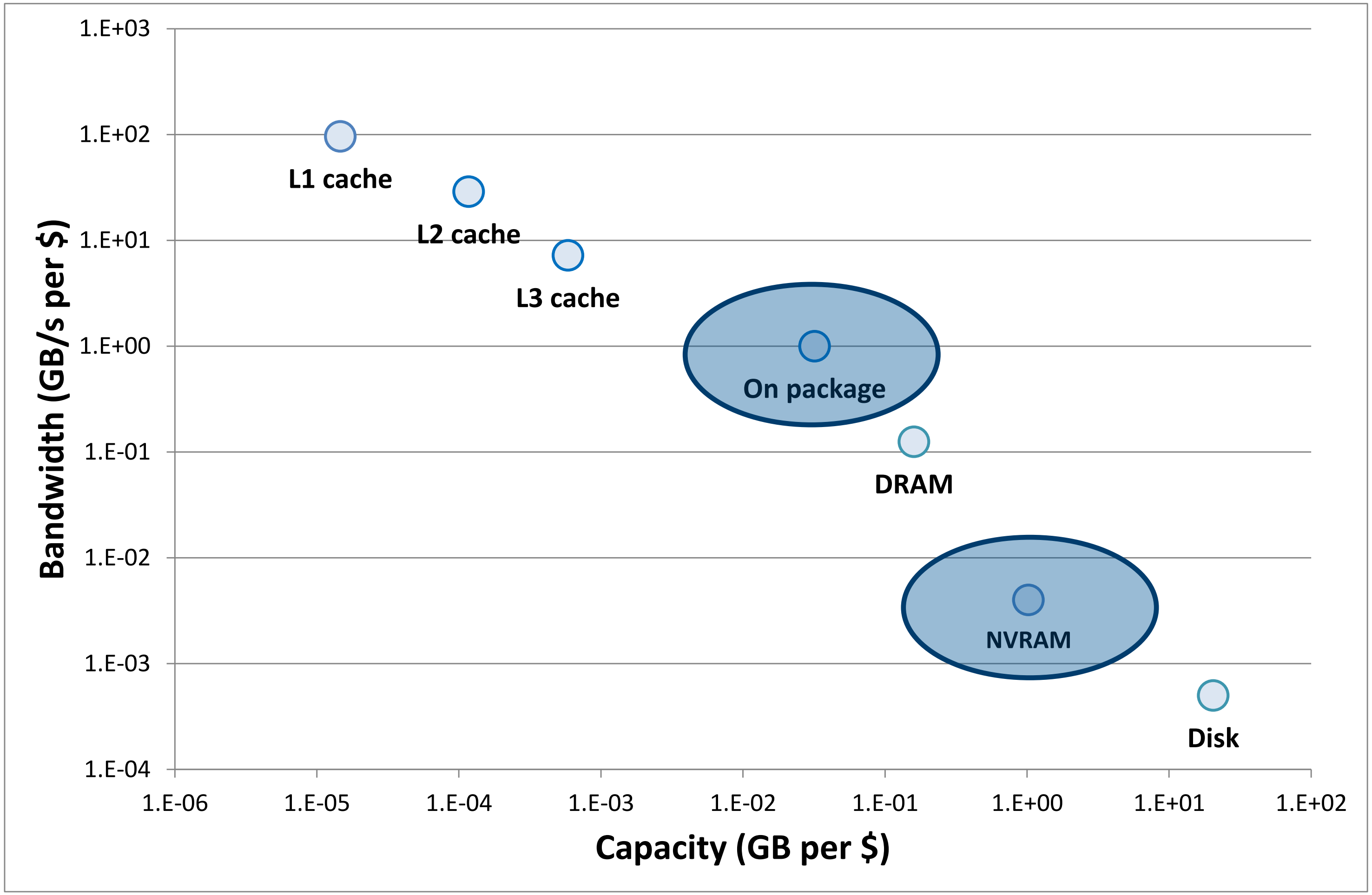
Memory hierarchy today



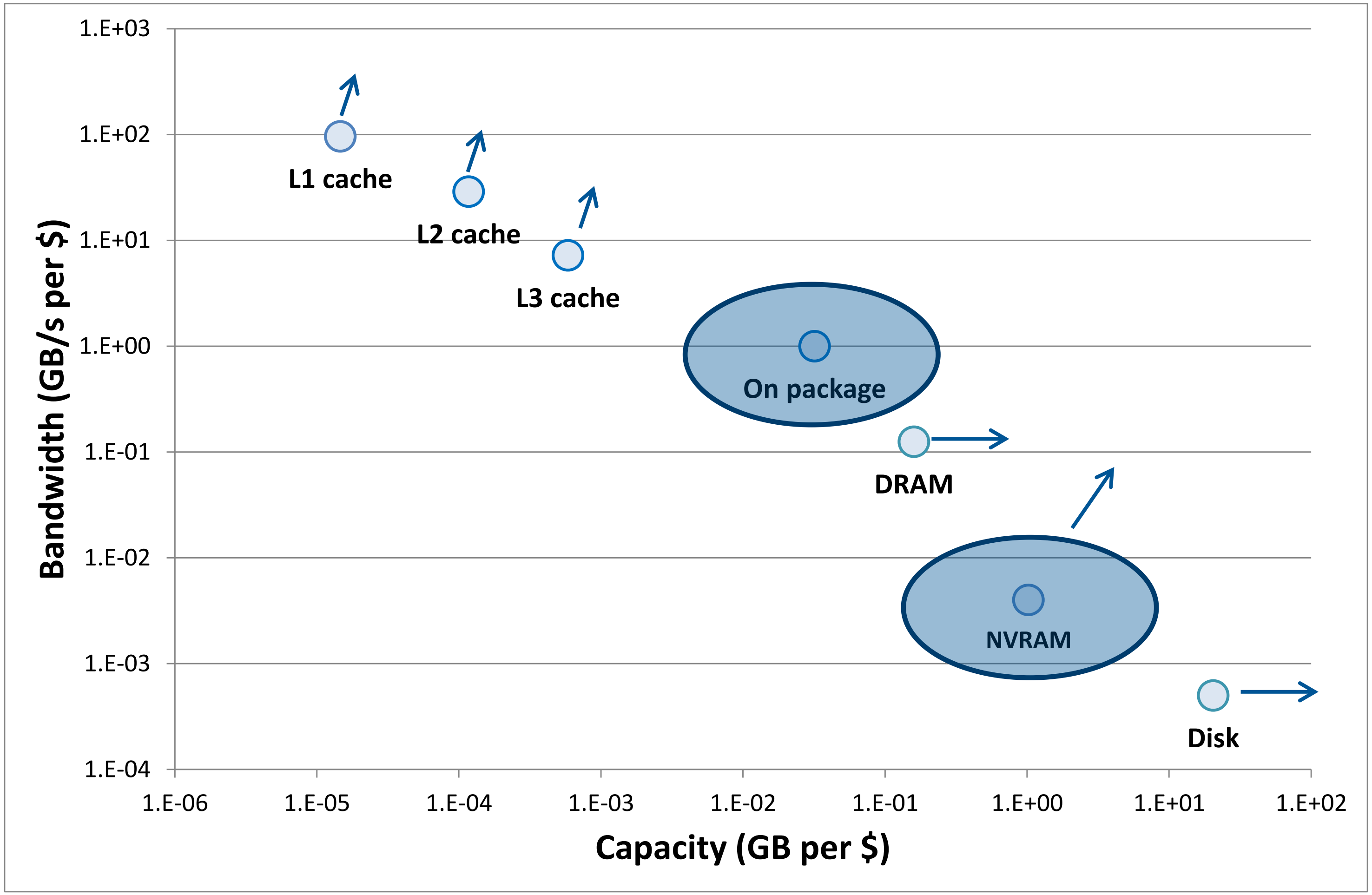
Additions to memory hierarchy



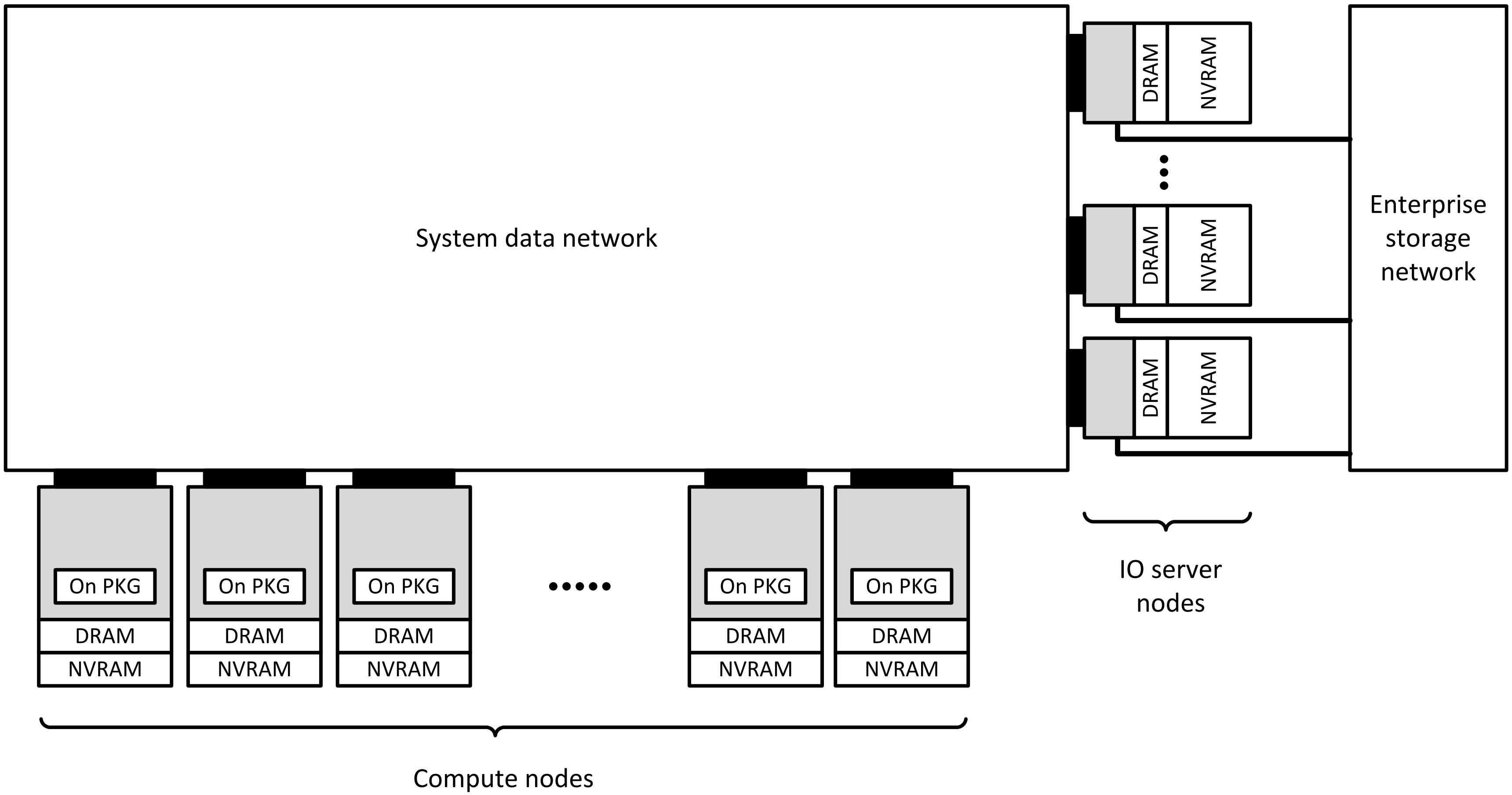
Additions to memory hierarchy



Memory hierarchy – trends



Deployment of new memory technologies



Use cases

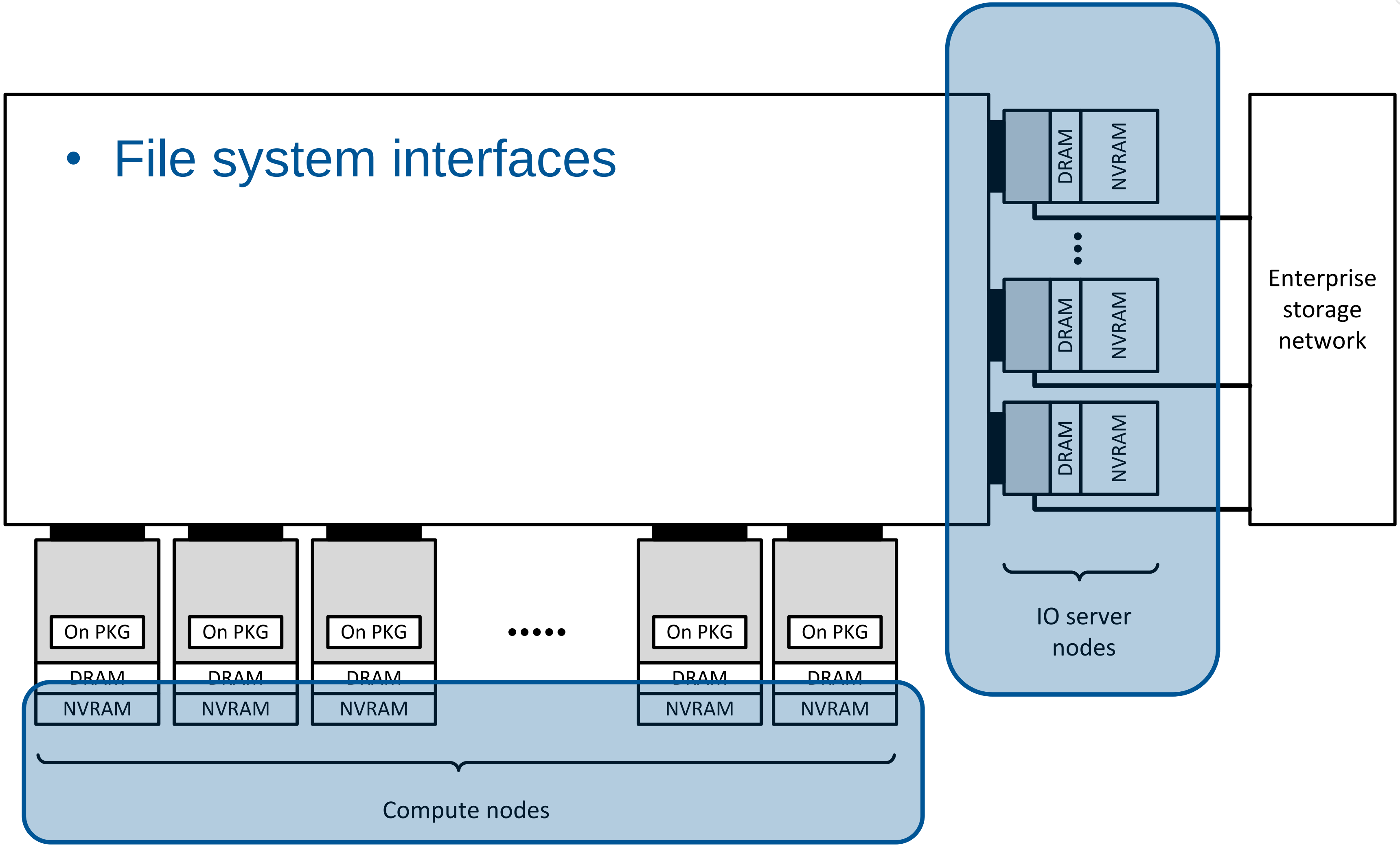
- **Improved workflow**
  - Bandwidth optimized storage
- **Improved analytics and visualization**
  - Tightly coupled access to the output of a simulation
  - On-the-fly analysis and steering
- **High memory applications**
  - A single application needs access to far more data than can reasonably be held in DRAM

Resource management use cases

- **Allocate X TB of NVRAM across a set of nodes for Y days**
  - Exclusive access, one job
  - Persistent access, set of jobs belonging to the same user
  - Shared access, data is accessed by any job with permission

Use of new memory technologies

- File system interfaces



File system interfaces

- **Separately managed file systems**
- **Cache of enterprise file system**
- **Many local file systems**
- **Active field**
  - PLFS, DAOS, DVS, ….

- **Use cases addressed**
  - Bandwidth optimised local storage
  - Some of the analysis and visualisation cases

Application use cases

- **Using each stage of memory as a cache**
  - On package memory as a cache of DRAM
  - DRAM as a cache of NVRAM
  - NVRAM as a cache of disk

- **Provides an easy way of using new technology**
- **Hides some of the complexities**
- **But what about applications that don't have a high degree of locality?**
  - They will need to have large numbers of requests in flight in order to hide the round trip latency.
  - As when accessing remote memory

Application use cases

- **Which data to hold at a particular level ?**
- **Which data to read from the level below and then discard?**

- **Explicit data movement primitives**
  - Looks a lot like remote data access
  - For example Put/Get

- **Compiler directives**
  - For example #pragma acc data

- **Area of significant interest for auto-tuning**
  - Instrument memory access patterns
  - Use this data to determine which data to hold at which level
  - Talk to my colleague Adrian Tate – his field of research

Parallel programming APIs?

- **Do nothing**
  - Each process allocates and manages its own NVRAM
  - Integrates with MPI + X programming model at process level
  - No direct access to extended memory of the whole job
  - Easy to get going
  - Hard to build in resiliency
  - Likely to result in lots of different solutions to the same problem

Parallel programming APIs?

- **Direct access**
  - Each process allocates its own NVRAM
  - Opens direct network access to it – e.g. MPI-3 RMA Window
  - Any process can access all of the NVRAM via RMA put/get operations
  - Reuses the existing client side API
  - Some system programming to do
  - Good fit for latency hiding
  - Hard to build in resiliency

Parallel programming APIs?

- **Distributed object access**
  - NVRAM allocated across some set of nodes – those in use by a job and/or I/O server nodes as well
  - Client API distributes requests over servers
  - Similar client API: put/get/sync
  - Layer over the same network API as MPI-3 RMA
  - Provides a way of hiding addressing and resiliency issues
  - Provides path to a wide range of analytics applications
- **Could provide a means for different applications to be accessing the same data**
  - Simulation code updating objects
  - Visualisation or analytics code consuming them

What else would I like to see?

- **Integrated work distribution mechanism**
  - Move the work to the data
- **Needs to be integrated into MPI**
  - Torsten has proposed this recently
  - Natural extension of MPI-3 remote accumulate

Conclusions

- **We expect our future systems to make extensive use of byte addressable NVRAM**
- **Important file system use cases**
  - Don't require major changes to existing applications
- **Easy to use caching mechanisms**
  - Will benefit some applications
  - Other applications will require explicit data movement
- **Interesting programming environment options:**
  - Direct access to the memory with explicit data movement
  - Distributed object access
- **Today's hardware and software can be used to prototype programming environment support**

Further information

PLFS	https://github.com/PLFS
DVS	http://docs.cray.com/books/S-0005-10/
Ramcloud	http://ramcloud.stanford.edu
NVSL	http://nvsl.ucsd.edu
OpenNVM	http://opennvm.github.io