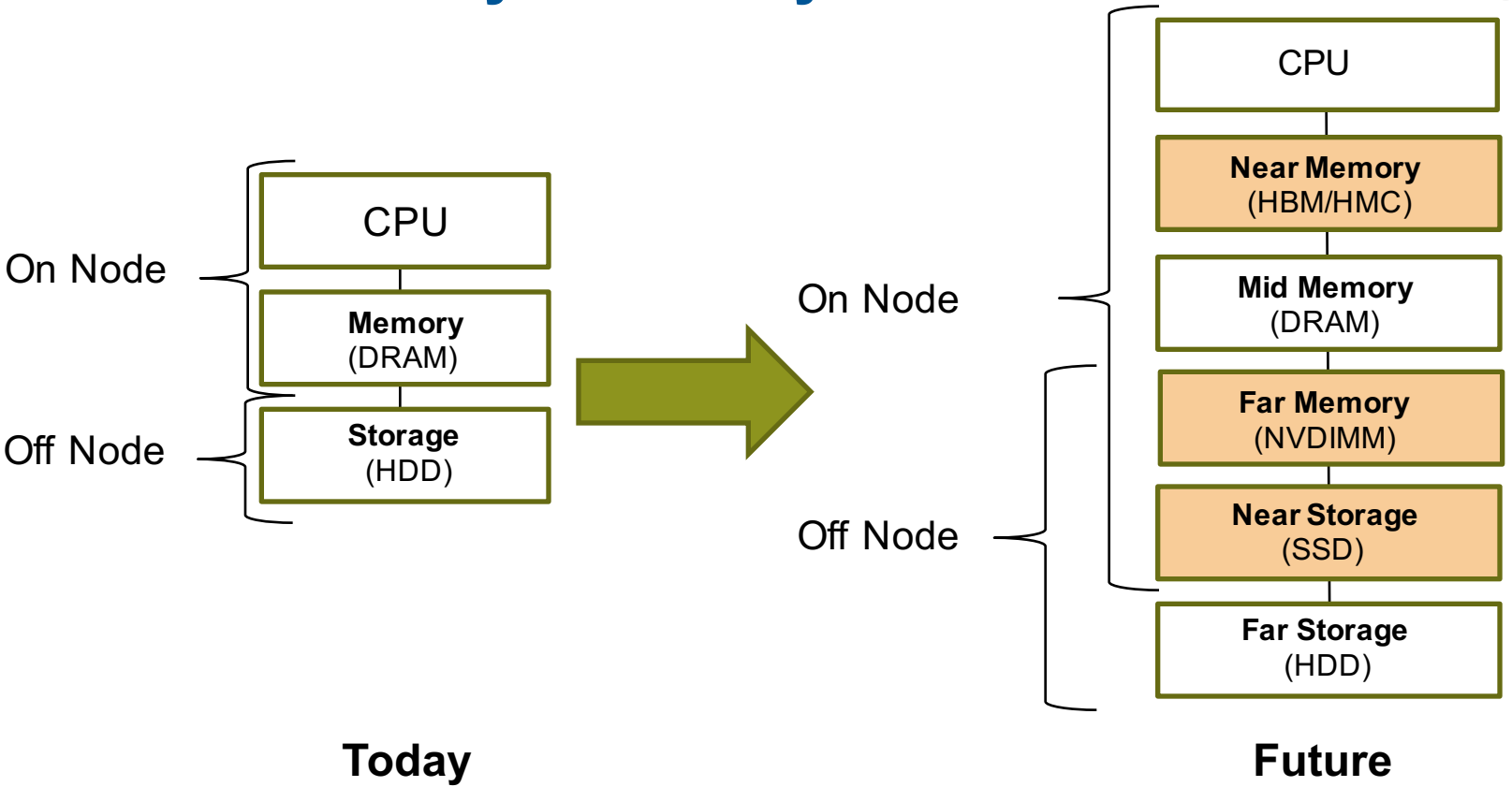
Managing the Memory Hierarchy

Larry Kaplan
Chief Software Architect
Cray Inc.

Now Where Did I Put That Data?

Larry Kaplan
Chief Software Architect
Cray Inc.

Abstract Memory Hierarchy

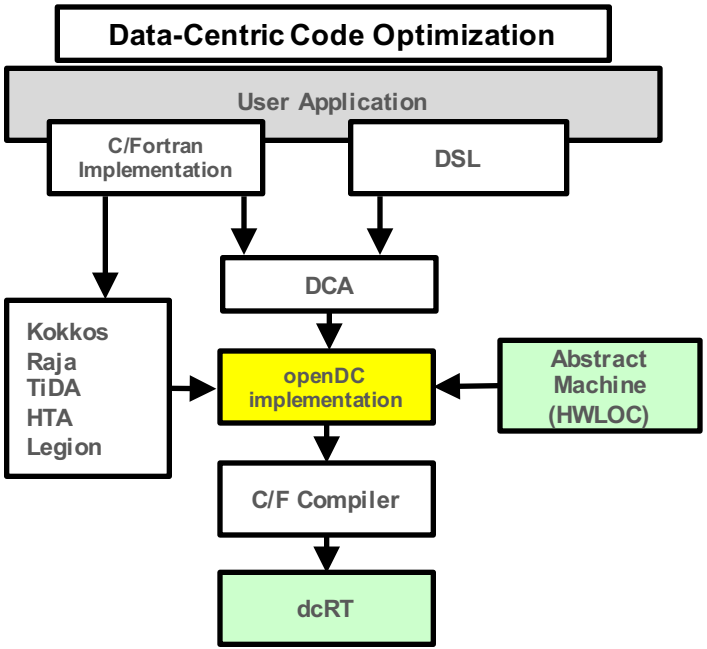


Data Management

- **Providing initial functionality today, developing more**
- **Will incorporate new technologies as they mature**
  - E.g. It's possible we'll have multiple levels of NVM in the hierarchy
- **Three levels of interest:**
  - On-node memory (on package, local bulk memory)
  - Network-attached NVM that is accessed as memory
  - Network attached NVM that is accessed via file system
- **At each level, want a dual approach**
  - APIs for users to manage and access data
    - APIs, directives, and tools for on-node memory management
    - Distributed object storage in NVM
    - DataWarp APIs
  - System software to automate placement and usage of the memory

Data Management in the Memory Hierarchy

- **Investigating programming model support for the various memories**
  - Largely concerned with placement and layout of data plus related work scheduling
- **Includes existing and future elements of memory hierarchy**
  - Caches, near/fast memory, bulk memory, remote memory, NVM
- **Community efforts making some progress**
  - Legion, Kokkos, RAJA, etc.
- **Cray is also investigating some proprietary techniques**
  - Expect synergy across efforts, potential for integration



Overall Hierarchy Management Strategy

- **OS provides the infrastructure**
  - Not well placed for decision making
  - Should expose detailed attributes to runtime
- **Programming model runtime aids or makes the decisions**
  - Has detailed application knowledge
  - Opportunity to hide details from programmer
- **Tools analyze usage to help with placement**
  - Especially for data not easily placed by application usage

Near Term Memory Hierarchy Plans

- **Developed initial set of directives for memory placement**
  - Agreement to work with Intel on standardization in OpenMP
- **Developing tools to advise programmer on placement**
  - Use performance counters and compiler instrumentation
- **Considering extensions to existing "placement" libraries**
- **NVRAM considerations are next**

CCE Support for High Bandwidth Memory

- **Support base Intel mechanisms**
  - Treat "memkind" API as a third-party library
  - Recognize "FASTMEM" attribute

- **Add directive (pragma) to support data allocation in HBM**
  - Support Fortran, C, and C++ and cover more use cases
  - The directive can be used on **both local and global variables**
  - The directive **can also be used on a statement**
    - Changes allocation routines (allocate, malloc, etc.) to use HBM
  - **If Clause** for dynamic control of directive
  - **Fallback Clause** to control behavior if allocation fails
  - Ideally will become part of a standard, possibly OpenMP

Directive for Existing Heap Allocations

```
!dir$ memory(attributes)
#pragma memory(attributes)
```

- **Appears prior to an allocation/deallocation statement**
  - Fortran: allocate
  - C/C++: malloc, calloc, realloc, posix_memalign, free
  - C++: new, delete, new[], delete[]
- **Directive on deallocation must match (C/C++ only)**
- **Converts normal allocation to high-bandwidth memory**
- **Initially "bandwidth" is the only allowed attribute**
  - Other attributes may be added in the future

Directive for Variable Declarations

```
!dir$ memory(attributes) list-of-vars
#pragma memory(attributes) list-of-vars
```

- **Specified at declaration of variable**
  - For global variables, directive must be visible for every use of global
  - Within type for allocatable members
- **Allowed on:**
  - Local and global variables
  - Scalars, structs, and arrays (fixed size and variable length)
  - Fortran allocatables (including members of derived types)
    - Memory allocated will use high bandwidth memory
- **Not allowed on:**
  - Dummy arguments, Fortran pointers, variables involved in equivalences, Coarray or UPC shared variables
  - Common blocks or variables within a common block

If Clause

```
!dir$ memory(attributes) if(expression)
#pragma memory(attributes) if(expression)
```

- **Dynamic control of directive**
- **For declarations**
  - Expression is evaluated when variable goes into scope
- **For heap allocations**
  - Expression is evaluated when directive is encountered
  - The expression must match on the deallocation (C/C++ only)

Fallback Clause

```
!dir$ memory(attributes) fallback
#pragma memory(attributes) fallback
```

- Controls behavior if allocation fails
- Default behavior: allocation fails
- Fallback behavior: allocation returns normal memory

Tool Support Plans for HBM

- **HBM summary info**
  - Memory use mode info (was cache or managed memory used?)
- **Program memory high water mark for sampling and tracing**
  - Break this down into DDR and HBM memory
- **Report active allocations at samples or during tracing at the function level**
- **Report overall program wallclock time, top time consuming routines, etc.**
- **Provide information on candidates for HBM placement**

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. 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.

Cray hardware and software products 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.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, URIKA and YARCDATA. The following are trademarks of Cray Inc.: ACE, APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, THREADSTORM. The following system family marks, and trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Other names and brands may be claimed as the property of others. Other product and service names mentioned herein are the trademarks of their respective owners.