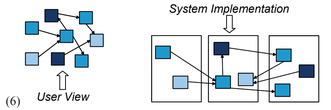
Charm++ on Cell

Parallel Programming Lab
University of Illinois Urbana-Champaign
David Kunzman, Gengbin Zheng, Eric Bohm, Laxmikant V. Kale

What is Charm++?

Charm++ is an message-driven paradigm developed by the Parallel Programming Lab (PPL) which uses messages to communicate between entities called Chares (C++ objects). The chares are then spread across the available processors. The runtime system automatically handles mapping of objects to processors, routing messages, load-balancing, automatic check-pointing, has associated tools for collecting performance data, etc.



(6)

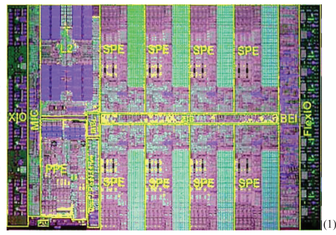
The current Charm++ Runtime System can run on a wide variety of platforms including SGI's Altix, IBM's BlueGene\L, clusters of workstations, and Alpha-based platforms.

Ongoing PPL Research Includes:

- Development of network-topology aware load-balancers
- Fault tolerance
- Dynamically changing the number of physical processors
- Multi-Cluster computing
- ParFUM: Parallel Framework for Unstructured Meshes
- Adaptive MPI (AMPI)

What is Cell?

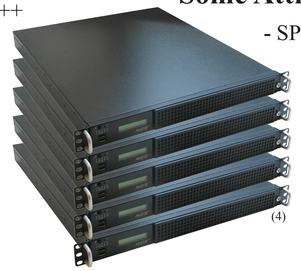
Jointly developed by IBM, Sony, and Toshiba, the Cell processor has 9 cores. One of the cores is a fairly standard PowerPC core. This "main core" is the called the Power Processing Element (PPE). The other 8 cores are called Synergistic Processing Elements (SPEs). A brief summary of the Cell's characteristics is below.



(1)

Some Attributes of Cell:

- SPEs run at same clock speed as PPE
- All SPE main memory accesses are done explicitly using DMA transactions
- SPE loads and stores can only access the local store (256KB, 6-cycle latency)
- Each SPE has 25.6 GFlop/s peak performance (single-precision @ 3.2 GHz)
- SPEs have two in-order pipelines: up to two instructions per clock
- 234M transistors (221mm²)
- Element Interconnect Bus (EIB): high-bandwidth from 74GB/s (worst-case) to 200GB/s (best-case)
- PPE is a 2-way SMT
- Both the PPE and SPE can initiate DMA transactions


(4)

One of the first products planned is the Dual Cell-Based Servers by Mercury Computer Systems. Each of the 1U units has a peak performance of about 400 GFlop/s each (single-precision). The five shown above can provide a total peak performance of 2TFlop/s.

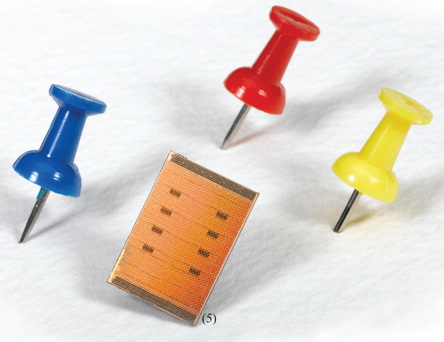
Initial Cell-based platforms include Sony's Playstation 3, Blade servers from IBM, and servers from Mercury Computer Systems.

What is NAMD?

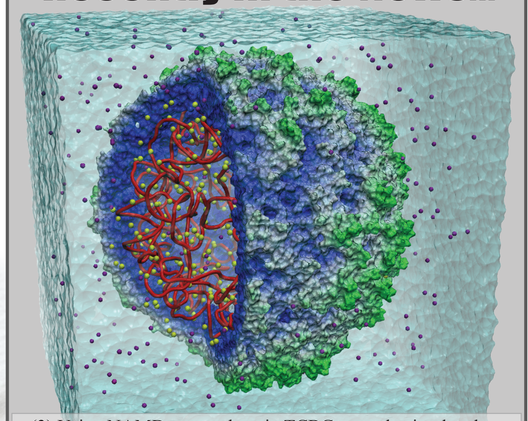
NAMD, a production molecular dynamics code, is the first Charm++ application that will be adapted to utilize the Cell. NAMD was developed by the Theoretical and Computational Biophysics Group (TCBG) at UIUC's Beckman Institute in collaboration with the Parallel Programming Lab. In 2002 it won the Gorden Bell Award and is currently being used around the world.

NAMD Highlights:

- 2002 Gorden Bell Award
- Scaled to 8000 Processors (BlueGene\L)
- File-Compatible with AMBER, CHARMM, X-PLOR
- Used by DOE National Labs


(5)

Recently in the News...



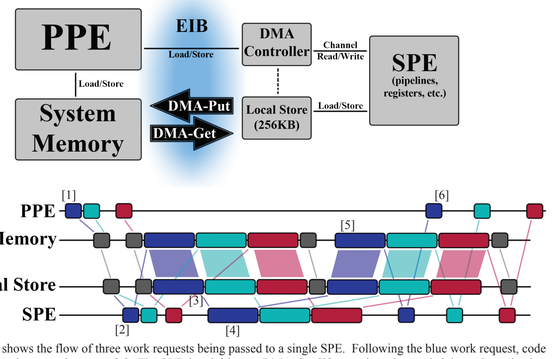
(3) Using NAMD, researchers in TCBG recently simulated an entire life form at the atomic level for the first time. They simulated the satellite tobacco mosaic virus. They used 256 processors on NCSA's SGI Altix to perform the 1,000,000 atom virus simulation for a total of 10ns (at 1.1ns per day).

Charm++ Complements Cell

- **Ease of Programming**
  - The nature of Cell makes it difficult to program
  - Charm++ programming model fits Cell well
- **Intelligent Message Scheduling**
  - Messages arriving for Chares are queued for execution
  - Scheduler peeks ahead in message queue to initiate DMA transfers of data/code that will be used by future messages while the current messages are still being processed
- **Load-Balancing**
  - Processing elements have different characteristics (SPE vs PPE)
  - Network-topology aware load-balancers (different communication costs between interconnect and EIB)
  - Application can use different load-balancers for different platforms
- **Portability**
  - Charm++ applications can already run on a variety of platforms
  - Will allow users to take advantage of Cell's power with little to no modification of existing code
- **Chares Encapsulate Data/Code (Locality)**
  - Arriving message is self-contained
  - Other data used by entry method is usually contained within the associated Chare

Charm++ on Cell so Far

We have created an *Offload API* that will allow Charm++ applications to take advantage of the Cell processor. Using the Offload API, the application can send *work requests* to the SPEs. Each of the SPEs has a simple scheduler running on it that will coordinate the execution of work requests along with moving the data needed by those work requests.



This figure shows the flow of three work requests being passed to a single SPE. Following the blue work request, code on the PPE makes a work request [1]. The SPE then initiates a DMA-Get [2] to retrieve the needed data. Once the data has arrived to the local store [3], the work request is executed on the SPE [4]. After completion, the data is once again placed back into system memory [5] and the PPE is notified that the work request has finished [6]. (Not to Scale)

Some simple example programs have already been written using Charm++ and the Offload API (a 2D Jacobi run is shown below).

(2) Screenshot of performance data generated using the Cell Simulator provided by IBM. This figure is a single iteration of a 2D stencil program written using Charm++ and the Offload API. This run utilizes all 8 SPEs on the Cell. Each SPE receives 8 evenly sized chunks of data.

Future Work

- **Application Development**: NAMD, Cosmology, etc.
- **Development of Load-Balancers** (considering several metrics)
  - Greatly varying communication costs depending on sending and receiving processing elements (over EIB, within node, and over the interconnect)
  - Varying capabilities of processing elements including memory size, ISA characteristics, etc.
- **Portability / Ease-of-Programming**
  - Modify Charmxi to allow offloadable keyword
  - Auto-generate code needed by Offload API based on user's code
- **Performance Analysis**: generate Projections data for SPEs
- **Special Purpose Hardware**: FPGAs, GPUs, etc.

(1) Image from: http://domino.research.ibm.com/comm/research.nsf/pages/r.arch.innovation.html
(2) Screenshot of Cell Simulator provided by IBM
(3) Image provided by the Theoretical and Computational Biophysics Group
(4) Image from Mercury Computer Systems Inc. Image Library (http://www.mc.com)
(5) Image from IBM Photo Gallery (http://www-03.ibm.com/press/us/en/photos.wss)
(6) Image from PPL Webpage (http://charm.cs.uiuc.edu/research/charm/)

Poster Created: 04/14/05

For More Information, Please Visit...
Charm++ : http://charm.cs.uiuc.edu
NAMD : http://www.ks.uiuc.edu