

## **Multi-Paradigm Run Time MPR for Multicore Systems**

*Xiaohong Qiu  
Research Computing UITS  
Indiana University, Bloomington IN*

Multicore architectures are bringing parallel computing to a broad range of applications with profound impact on hardware, systems software and applications [1-3]. The programming models and runtime that will be used on multicore architectures are the subject of substantial academic and industry research and development as they must bridge between current commercial desktop and server systems, commercial parallel databases, distributed Grid environments and the massively parallel supercomputers largely aimed at science and engineering [4]. Intel [5] has examined classes of possible future desktop applications which they term RMS – Recognition, Mining and Synthesis. This can be illustrated by a sophisticated data mining application that first accesses a parallel database and then runs analysis algorithms including perhaps a multithreaded branch and bound search, and a SVM Support Vector Machine built on parallel linear algebra followed by sophisticated visualization. This composite application would need to be controlled by a coarse grain executive similar to Grid workflow [6] or Google MapReduce [7]. The individual data mining filters could use either the dynamic thread parallelism appropriate for a combinatorial search algorithm or an MPI style messaging for parallel linear algebra used in the SVM filter. Further we would like this job to run efficiently and seamlessly either internally to a single CPU or across a tightly coupled cluster or distributed Grid. In this white paper we highlight a small part of the multicore puzzle – namely what could be the runtime that could span these different environments and different platforms that would be used by the heterogeneous composite applications that could be common on future multicore applications for personal, commercial or research use.

We suggest that it would be very important to have a common run time to support the key parallel models so that we can integrate modules of different styles. Four low level run time scenarios are MPI collective messaging, asynchronous threading; coarse grain functional parallelism or workflow and discrete event simulation. We have looked at the first three paradigms in a recent paper

(<http://grids.ucs.indiana.edu/ptliupages/publications/CCRApril16open.pdf>) and shown that CCR is a possible messaging substrate with good performance across the different paradigms. CCR developed in Microsoft Research is a runtime [8-9] designed for robotics applications [10] but also investigated [11] as a general programming paradigm. There are obviously other choices and for understanding how to develop and compare such runtimes, we suggest that it would be useful to discuss both the API's and performance requirements for multi-paradigm runtimes. In particular, we wonder if it would be useful to define a simpler core MPI so that would be easier to incorporate into the set of multi-paradigm API's. For example CCR has a few core capabilities and one builds additional functionality with a convenient C# framework. One could then only put the ability to do MPI style collective loosely synchronous communication into MPR (Multi Paradigm Run Time) and allow extensions. Our work suggests an MPR is feasible but it needs more work to evaluate other requirements (such as OpenMP, HPCS languages [12], interoperation with transactional memory and locks) and other languages (C, Java and functional languages seem especially important) that could constrain MPR.

## REFERENCES

1. Jack Dongarra Editor *The Promise and Perils of the Coming Multicore Revolution and Its Impact*, CTWatch Quarterly Vol 3 No. 1 February 07,  
<http://www.ctwatch.org/quarterly/archives/february-2007>
2. Herb Sutter, *The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software*, Dr. Dobbs's Journal, 30(3), March 2005.
3. Annotated list of multicore Internet sites <http://www.connotea.org/user/crmc/>
4. Geoffrey Fox tutorial at Microsoft Research *Parallel Computing 2007: Lessons for a Multicore Future from the Past* February 26 to March 1 2007  
<http://grids.ucs.indiana.edu/ptliupages/presentations/PC2007/index.html>
5. Pradeep Dubey *Teraflops for the Masses: Killer Apps of Tomorrow* Workshop on Edge Computing Using New Commodity Architectures, UNC 23 May 2006  
<http://gamma.cs.unc.edu/EDGE/SLIDES/dubey.pdf>

6. Dennis Gannon and Geoffrey Fox, *Workflow in Grid Systems* Concurrency and Computation: Practice & Experience 18 (10), 1009-19 (Aug 2006), Editorial of special issue prepared from GGF10 Berlin <http://grids.ucs.indiana.edu/ptliupages/publications/Workflow-overview.pdf>
7. Jeffrey Dean and Sanjay Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*, OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December 2004 <http://labs.google.com/papers/mapreduce.html>
8. “Concurrency Runtime: An Asynchronous Messaging Library for C# 2.0” George Chrysanthakopoulos Channel9 Wiki Microsoft <http://channel9.msdn.com/wiki/default.aspx/Channel9.ConcurrencyRuntime>
9. “Concurrent Affairs: Concurrent Affairs: Concurrency and Coordination Runtime”, Jeffrey Richter Microsoft <http://msdn.microsoft.com/msdnmag/issues/06/09/ConcurrentAffairs/default.aspx>
10. Microsoft Robotics Studio is a Windows-based environment that provides easy creation of robotics applications across a wide variety of hardware. It includes end-to-end Robotics Development Platform, lightweight service-oriented runtime, and a scalable and extensible platform. For details, see <http://msdn.microsoft.com/robotics/> with tutorials at [http://msdn.microsoft.com/robotics/learn/On\\_Demand/default.aspx](http://msdn.microsoft.com/robotics/learn/On_Demand/default.aspx)
11. Georgio Chrysanthakopoulos and Satnam Singh “An Asynchronous Messaging Library for C#”, Synchronization and Concurrency in Object-Oriented Languages (SCOOL) at OOPSLA October 2005 Workshop, San Diego, CA. <http://urresearch.rochester.edu/handle/1802/2105>
12. Internet Resource for HPCS Languages [http://crd.lbl.gov/~parry/hpcs\\_resources.html](http://crd.lbl.gov/~parry/hpcs_resources.html)