Project Proposal

A Decoupled Execution Paradigm Programming Model

CS550 – Advanced Operating System

September 12th 2016

Anthony Kougkas

A20307881

1. Introduction

Many scientific applications running in HPC systems deal with large amount of data. On the other hand, the computation power of current multicore/manycore architectures has followed a faster trend compared to data access performance improvement (latency and bandwidth). This gap is very unlikely to be overcome in the near future. Thus, accessing large amount of data becomes a bottleneck for many applications [1]. This movement of large amount of data constantly in an application can cause severe degradation to the overall performance [2]. One of the promising architectures is the "Decoupled execution paradigm"(DEP) [3], where the application is decoupled into computive-intensive and data-intensive phases and each are directed to the appropriate capable nodes The ability to address the challenges when programming with DEP architectures are covered by this projects proposed solution which includes the design and implementation of a novel programming model called DEPM.

2. Background Information

DEP is the first paradigm enabling users to identify and handle data-intensive operations separately. It can significantly reduce costly data movement and is better than the existing execution paradigms for data-intensive applications. Data nodes can provide simple data forwarding without any data size reduction, but the idea behind data nodes is to let the data nodes conduct the decoupled data-intensive operations and optimizations to reduce the data size and movement. Figure 1 depicts this architecture. Active storage [4], [5], [6] leverages the computing capability of storage nodes and performs certain computation to reduce the bandwidth requirement between storage and compute nodes. Active disks [7] and smart disks integrate a processing unit within disk storage devices and offload computations to embedded processing unit.

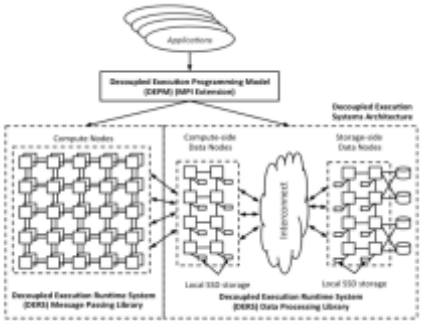

Figure 1, Decoupled Execution Paradigm for Data-Intensive Applications

3. Problem Statement

Even though new system architectures for data-intensive computing are proposed, scientists still use the old-fashion compute-centric parallel programming models to program these system. These programming models primarily focus on the memory abstractions and communication mechanism among processes. I/O is treated as a peripheral activity and often a separate phase in these programming models, which is often achieved through a subset of interfaces such as MPI-IO. This mismatch of programming data-intensive applications with compute-centric approaches makes the programming difficult and leads to error prone and complex codes.

4. Related Work

There have been some development of advanced I/O libraries, such as Hierarchical Data Format (HDF), Parallel netCDF (pnetCDF) and Adaptable IO System (ADIOS), that all provide high level abstractions, map the abstractions to the I/O and try to complement parallel programming models, such as Message Passing Interface (MPI) [8] and others, by managing I/O activities. MapRecude [9] as mentioned before, is another example of a programming model aiming to ease data-intensive applications. OpenMP [10] is an API that supports multi-platform shared memory multiprocessing programming on most processor architectures and operating systems. Much like this, DEPM will provide the simple and flexible interface for developing DEP applications running on this new decoupled execution framework.

5. Proposed Solution

In this study, we are proposing a new programming model that will facilitate the decoupled execution paradigm and help developers code easily and efficiently. The DEPM (DEP programming model) uses a source-to-source compiler that can translate DEPM code to the traditional HPC code. DEPM is inherently tied not only to the underlying hardware infrastructure but also to the requirements of applications and libraries. Our model exposes features of the runtime software layers and also effectively presents to the software developer traditional and

new software development tools. We provide some evaluation of more complex applications that use DEP and demonstrate the potential of this new execution model through the use of our DEPM. The proposed solution comprises by:

- Implementing the DEP programming model (C++ code)
- Evaluating the efficiency of the DEPM
- Evaluating complex applications using DEPM.

6. Evaluation

The evaluation of this project is comprised by the following:

- Comparison of programming effort by use case (number of lines of codes) in DEP programming model vs. naive decoupling code vs. code in existing architecture
- Performance comparison of DEP architecture code with existing architecture's implementation
- For the sake of these comparisons we use three different use cases (scenarios):
    - Sorting
    - Min/Max/Avg as proposed in the original DEP paper
    - Graph500

  Measurements will be done for total execution time (sec), I/O performance (bandwidth) and programming effort (LOC). The testbed machine that will be used for those tests is the HEC cluster from SCS lab in IIT.

7. Conclusions

A good programming model expresses its functionality in a way that matches how the developer wants to think about it. A good programming model doesn't just expose internal structures—it exposes its functionality at a higher level of abstraction so that the developer can concentrate on what he/she wants to do and not on how one has to accomplish a simple task.
Our programming model will be designed with common scenarios in mind. What will developers typically want to do with our component? Which features are more important for advanced users and which features should be very easy to use? Note that the focus in this project will be on the features and not on the internal objects, data structures, or functions. Our programming model will be designed with common scenarios in mind. What will developers typically want to do with our component? Which features are more important for advanced users and which features should be very easy to use? Note that the focus in this project will be on the features and not on the internal objects, data structures, or functions. We strongly believe that with little change in legacy data-intensive code, our DEP programming model can achieve significant benefits in the performance.

8. Additional Resources

8.a) Timeline

Week	*Task*
1	Study relevant papers, gather more info, compile all together
2	Use case algorithmic design
3	Code the new programming model
4	Present intermediate report (midterm)
5	Code the new programming model / debug
6	Do the experimental evaluation
7	Write the final report
8	Write the final presentation

8.b) Deliverables

- One final report in PDF form.
- One final Powerpoint presentation.
- Source code.

References

[1] A. Choudhary, W. Liao, K. Gao, A. Nisar, R. Ross, R. Thakur, and R. Latham, "Scalable I/O and analytics," *J. Phys. Conf. Ser.*, vol. 180, p. 012048, Jul. 2009.

[2] P. Roadmap, J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J.- Claude, D. Barkai, J.-Y. Berthou, T. Boku, B. Braunschweig, B. Chapman, X. Chi, A. Choudhary, S. Dosanjh, T. Dunning, S. Fiore, A. Geist, B. Gropp, R. Harrison, M. Hereld, M. Heroux, K. Hotta, Y. Ishikawa, Z. Jin, F. Johnson, S. Kale, R. Kenway, D. Keyes, B. Kramer, J. Labarta, A. Lichnewsky, T. Lippert, B. Lucas, S. Matsuoka, P. Messina, P. Michielse, B. Mohr, M. Mueller, W. Nagel, H. Nakashima, M. E. Papka, D. Reed, M. Sato, E. Seidel, J. Shalf, D. Skinner, M. Snir, T. Sterling, R. Stevens, F. Streitz, B. Sugar, S. Sumimoto, W. Tang, J. Taylor, R. Thakur, A. Trefethen, and M. Valero, "The International Exascale Software."

[3] Y. Chen, C. Chen, X. H. Sun, W. D. Gropp, and R. Thakur, "A decoupled execution paradigm for data-intensive high-end computing," *Proc. - 2012 IEEE Int. Conf. Clust. Comput. Clust. 2012*, pp. 200–208, 2012.

[4] C. Chen and Y. Chen, "Dynamic active storage for high performance I/O," *Proc. Int. Conf. Parallel Process.*, pp. 379–388, 2012.

[5] J. Piernas, J. Nieplocha, and E. J. Felix, "Evaluation of active storage strategies for the lustre parallel file system," *Proc. 2007 ACM/IEEE Conf. Supercomput. (SC '07)*, no. 1, 2007.

[6] S. W. Son, S. Lang, P. Carns, R. Ross, R. Thakur, B. Ozisikyilmaz, P. Kumar, W. K. Liao, and A. Choudhary, "Enabling active storage on parallel I/O software stacks," *2010 IEEE 26th Symp. Mass Storage Syst. Technol. MSST2010*, 2010.

[7] J. Saltz, "Active Disks : Programming Model , Algorithms and Evaluation."

[8] S. Kumar and M. Blocksome, "Scalable MPI-3 . 0 RMA on the Blue Gene / Q Supercomputer," pp. 7–12.

[9] J. Dean and S. Ghemawat, "MapReduce : Simplified Data Processing on Large Clusters," *Commun. ACM*, vol. 51, no. 1, pp. 1–13, 2008.

[10] A. Basumallik, S.-J. Min, and R. Eigenmann, "Programming Distributed Memory Systems Using OpenMP," *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS 2007)*, pp. 1–8, 2007.