# Characterizing the Software Development Process of High Performance Computing Systems

Michael Paulding
*Collaborative Software Development Laboratory*
*Department of Information and Computer Sciences*
*University of Hawai'i*
*Honolulu, HI 96822*
*mpauldin@hawaii.edu*

## Abstract

*High performance computing systems are being applied to an increasingly wide variety of domains and are also becoming radically less expensive. However, the Defense Advanced Research Projects Agency (DARPA) High Productivity Computing Systems (HPCS) program notes that dramatic increases in low-level HPCS benchmarks of execution speed do not necessarily translate into increased development productivity. Essentially, the bottleneck in HPCS applications is moving from hardware limitations to software engineering limitations.*

*In addition, HPCS development adds another layer of complexity over what is imposed by traditional software engineering. It is often thought that the time devoted to parallelizing and optimizing code consumes a majority of the total development effort. However, empirical, public domain estimations of time allocation parallel and sequential development time are limited or non-existent.*

*In this research, a paradigm is proposed to enable a developer or project manager to understand and characterize his HPCS development. The paradigm follows several codependent processes that provide a broad perspective of the development. One process of this paradigm is to assess progress through milestone tests, each of which is a unit of functionality of the application that, when combined together, form the complete system. Another process is the tracking of the degree of parallelism in source code, or a measure of parallel constructs (routine calls and global variables) that appear in the files over time. Finally, the parallel performance of the application will be chronologically recorded.*

*In order to accomplish the objectives of this research, a four part thesis is proposed:*

1. *The Degree of Parallelism (DOP) analysis is useful and accurate for measuring the level of parallelism in HPCS source code.*
2. *The Progress Assessment through Milestone Tests (PAMT) analysis is useful and accurate for assessing progress in the development of an HPCS application.*
3. *The Parallel Performance Analysis (PPA) provides a useful and accurate means for assessing the parallel performance of an HPCS application.*
4. *The three proposed analyses, DOP, PAMT, and PPA can be combined to provide a paradigm for understanding and characterizing HPCS development.*

*To complete this research in a timely fashion, I will adhere to the following timeline:*

1. *January 2005 - implementation of software system for characterizing HPCS development.*
2. *March 2005 Evaluation of system under academic and industry settings.*
3. *April 2005 - Submission and defense of thesis for M.S. degree.*

## 1. Evaluation

In order to evaluate the claims proposed by this research, it is first necessary to understand the proposed benefits of the research. Once the intended benefits have been established, the strategy for evaluation of the benefits will be discussed in detail.

To begin with, the focus of this research, as stated in the title, is to characterize the software development process of high performance computing systems. In order to accomplish the characterization, this research proposes to automatically and unobtrusively provide analyses that encompass HPCS software development. These analyses include:

1. **Degree of Parallelism (DOP) of Source Code**
2. **Progress Assessment through Milestone Tests**
3. **Parallel Performance Analysis**

These analyses are independent of each other and retain loose coupling, but each is necessary to perform a full characterization of the HPCS software development process.

### 1.1 Degree of Parallelism (DOP) of Source Code

Determining the DOP of source code is an important component of understanding and characterizing HPCS development.

As a refresher, the DOP for source code functions as a binary relationship. If a source file contains one or more parallel constructs, then that file is assigned a DOP = 1. A parallel construct is defined as a routine or global variable that is used for message passing among processors in a parallel system. Commonly used libraries implementing message passing are the Message Passing Interface (MPI) and Open Message Passing (OpenMP), so specific examples of parallel constructs involving MPI or OpenMP routines or global variables. Files containing at least one parallel construct are therefore termed "parallel" files.

In contrast, some files in an HPCS application are completely absent of parallel constructs and are, in turn, assigned a DOP = 0. These files typically involve units of computation that do not require enough resources or execution time to warrant parallelization. For example, computing the factorial of a single integer is likely to be done sequentially, whereas a factorial problem containing a dataset of thousands of integers is likely to be distributed and computed in parallel. In essence, a source file containing no parallel constructs is termed a "sequential" file.

The metric for degree of parallelism in source code provides a plethora of benefits and insights for the HPCS developer and project manager. To start with, the development of an HPCS application typically consists of two processes: the implementation of the system functionality (sequential part) and the optimization and parallelization of that functionality (parallel part). It is often hypothesized among HPCS developers what portion of time is devoted to parallelization, with some estimates as high as 90there is no concrete mechanism to measure the ratio.

In addition to providing a tangible value representing the ratio of sequential and parallel files in a project, this metric provides added value when coupled with the active time metric. A developer or project manager can use this analysis to look at historical and in-progress data to see how much time was spent parallelizing code. If the value they observe seems disproportionate to previous project data or to their mental model, value is added by inspecting the parallel files. Consider the case in which one or more developers has spent a large portion of the total active time on parallelization. This set of developers (or their project manager(s)) can look at the parallel files edited during that time interval and identify what parallel constructs or optimization strategies caused the time drain. Once identified, a project manager can record this data and try to divert future teams from taking these resource draining approaches and try new strategies to obtain better productivity. In addition, using the archived data, other development teams can learn about past bottlenecks before embarking on a new development endeavour.

In order to evaluate whether this metric provides a useful and accurate value for the degree of parallelism in source code, I will perform the following:

1. Incorporate the DOP tool into the daily build of

HPCS developers, sample size 5 (is this enough?)

2. Create Hackystat accounts for each of the developers to permit them access to their own data.

3. Gather data on Hackystat over a period of 4-6 weeks, allowing developers to view and interpret their in-process data.

4. Distribute a survey to the developers and their project manager(s) to evaluate whether the DOP analysis was useful and accurate for measuring the level of parallelism in their source code.

### 1.2 Progress Assessment through Milestone Tests (PAMT)

The PAMT analysis provides a means for a developer or program manager to understand the progress made during the development of an HPCS application.

To understand this analysis, it is first necessary to provide a concrete definition for the term milestone test. Milestone tests are, in the simplest form, a set of tests that fully describe the functionality required in the HPCS application. For example, in the Truss PBB, the set of milestone tests would include:

MT-1) The application can successfully generate a topology connecting the attachment points on a wall to the load bearing point in free space, using a variable number of joints.

MT-2) The application can successfully assign a geometry to a previously defined topology. Geometry assignment includes determining the type of member connecting two joints in the truss. In the Truss PBB, the two options are cable or strut.

MT-3) Given a topology, the application must assert that the distance between joints does not exceed the deformational constraints of steel. In short, MT-3 will pass only if each distance between joints does not exceed a fixed limit.

MT-4) Given a truss topology and geometry, the application can compute a set of forces acting on each joint in the truss.

MT-5) Given a set of forces acting on a joint, the application can verify that the net force is zero.

MT-6) Given the net force on each joint, the application can determine whether the truss is in equilibrium.

MT-7) Given a truss topology and geometry, the application can calculate the mass of each member.

MT-8) Given a truss topology and geometry, the application can calculate the total mass of the truss, providing a potential solution for the Truss PBB problem.

It is important to note that each of these milestone tests are independent of all others, but each of the tests must be satisfied before the application can be considered complete. Therefore, developers could follow many different permutations of these tests to achieve overall completion. However, even though the paths developers take to implement the tasks may vary significantly, their overall progress through the application is comparable.

For example, consider the case in which the program manager decides to weight each of the milestone tests equally, giving each test a value of 12.5. Two developers implementing the application independently can then be evaluated jointly based on the score of tests they have implemented. Again, both developers will be finished when their score equals 100. On the other hand, if a program manager decides to assign milestones with differing weights, one developer may have implemented more milestone tests, but have a lower score than the other developer. Regardless, the overall progress of these developers is still simply comparable.

In order to evaluate whether the PAMT analysis is useful and accurate for assessing progress in a HPCS application, I will perform the following:

1. Incorporate the sensors and tools needed to capture the milestone test data (e.g. pass/fail) for a set of HPCS developers, sample size 5 (is this enough?)

2. Create Hackystat accounts for each of the developers to permit them access to their own data.

3. Gather data on Hackystat over a period of 4-6 weeks, allowing developers to view and interpret their in-process data and total milestone test score.

4. Distribute a survey to the developers and their project manager(s) to evaluate whether the PAMT analysis was useful and accurate for assessing progress.

### 1.3 Parallel Performance Analysis (PPA)

Parallel performance analysis (PPA) provides critical measurements for how well a HPCS application is performing and how well it is been parallelized and optimized.

PPA provides 5 universal analyses for quantifying performance characteristics of parallel software in a generic and portable manner. Each of the 5 analyses is briefly described below:

1. **Speedup** - Speedup is used as a measure to indicate the degree of speed gain in a parallel computation. This analysis is generally measured in the formula S(n) = T(1)/T(n), where n is the number of processors used and T(n) is the execution time used by n processors. However, the speedup analysis formula in particular has several variations depending on how a parallel algorithm is implemented.

2. **Efficiency** - Efficiency is used as a measure to indicate the useful portion of the work performed by n processors. This analysis is generally measured by the formula E(n) = S(n)/n.

3. **Redundancy** - Redundancy is used as a measure to indicate the extent of workload increase. Conceptually it can be though of as how many operations are duplicated with the addition of a processor. This analysis is generally measured by the formula R(n) = V(n)/V(1), where V(n) is the number of unit operations performed on n processors.

4. **Utilization** - Utilization is used as a measure to indicate the extent to which resources are utilized during a parallel computation. This analysis is generally measured by the formula U(n) = R(n)E(n).

5. **Quality** - Quality is used as a measure for combining the effects of speedup, efficiency and redundancy to assess the relative merit of a parallel computation. In effect, the quality analysis serves as the most succinct and direct comparison between two parallel implementations of the same problem.

Parallel performance analysis provides useful, if not critical, information to a HPCS developer or project manager. For example, PPA provides instant feedback for whether a change made to the system further optimized it or slowed it down. This information is available in process and development teams can make adjustments as necessary before the project deadline.

In addition, PPA couples well with the DOP and PAMT analyses. Specifically, by combining these three analyses, a developer or project manager is enabled to understand the impact on performance when functionality is increased or decreased (e.g. more milestone tests pass or fail) and when the degree of parallelism of the source code is increased or decreased (e.g. more time is spent parallelizing or optimizing).

1. Incorporate the sensors and tools needed to capture the parallel performance analysis data (e.g. the 5 universal performance quantifiers) for a set of HPCS developers, sample size 5 (is this enough?)

2. Create Hackystat accounts for each of the developers to permit them access to their own data.

3. Gather performance data on Hackystat over a period of 4-6 weeks, allowing developers to view and interpret their in-process data and performance values.

4. Distribute a survey to the developers and their project manager(s) to evaluate whether the PPA analysis was useful and accurate for assessing parallel performance of their HPCS application.