

January 01, 2007

## Semi-automated parallelization using Star-P

Dana Schaa

*Northeastern University, Honor Society Of Phi Kappa Phi*

David Kaeli

*Northeastern University*

Alan Edelman

*Interactive Supercomputing*

---

### Recommended Citation

Schaa, Dana; Kaeli, David; and Edelman, Alan, "Semi-automated parallelization using Star-P" (2007). *Research Thrust R3 Presentations*. Paper 15. <http://hdl.handle.net/2047/d10009164>

This work is available open access, hosted by Northeastern University.



# "SEMI-AUTOMATED PARALLELISM USING STAR-P"

Dana Schaa<sup>1</sup>, David Kaeli<sup>1</sup> and Alan Edelman<sup>2</sup>

<sup>1</sup> Department of Electrical and Computer Engineering  
Northeastern University, Boston, MA  
{dschaa, kaeli}@ece.neu.edu

<sup>2</sup> Interactive Supercomputing  
32 Beaver St, Waltham, MA  
edelman@interactivesupercomputing.com



## Objective

The purpose of this research is to determine the effectiveness of Star-P as a tool for exploiting parallelism in high-performance scientific computing, and eventually contributing to its functionality—perhaps by porting it to run on other high performance platforms such as graphics processors.

The most attractive feature of Star-P is that it allows researchers and scientists to produce code which is capable of running on high-performance parallel machines, without having to consider concepts related to parallelism or even leave the familiarity of their Matlab environment.

## What is Star-P?

Star-P (\*p) is an interactive parallel computing platform that gives researchers and scientists the ability to create parallel programs capable of running on high-performance architectures without requiring any parallel programming knowledge. Once installed, the Star-P client connects a desktop application, such as Matlab, to a parallel server, and outsources the most computationally intensive operations to it.

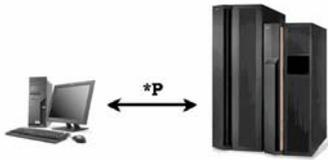


Figure 1: Connecting workstations to high-performance servers

Star-P allows the user to exploit data parallelism using the \*p operator, and task parallelism using the ppeval function.

More information about Star-P can be found at:  
<<http://www.interactivesupercomputing.com>>

## State of the Art

There are a large number of parallel-Matlab applications currently in existence. Each provides a semi-automated approach and has a different take on what gets parallelized and how it's done. Each approach has its strengths and weaknesses.

Star-P developers have created a subset of existing Matlab functions that operate on a new Star-P data type. Tagging Matlab with \*p converts them to the Star-P data type which can inherently take advantage of parallelism. In this way, existing code can be modified quite easily to run in parallel, or can still be executed serially if desired. Also, the distributed Star-P approach allows the user to worry less about memory limitations because data can be created across the memories of each node on the high performance server.

Other current options for parallelization include lower level approaches such as MPI, OpenMP, or grid computing. These approaches are likely to achieve larger speedups, but require a skilled parallel programmer to implement them effectively and subsequently have a much larger and more costly development time.

## Acknowledgement

This project is supported by an NSF STTR Phase I Award titled "Commercial Grade Automatic and Manual Parallelization and Performance Tool" NSF STTR Phase I Award No. 0638034, and an NSF MRI Award titled "MRI/Acq: Enabling Research on Terabyte-Scale Datasets," NSF MRI Award No. 0619616.

## Technical Approach

Our approach to determining the effectiveness of Star-P is to parallelize existing scientific applications. The procedure and goals are as follows:

- Use a *diary method* to record time spent on different aspects of parallelization.
- From the results of time spent, determine if there are any tasks which are difficult for the programmer when using Star-P.
- Using comparative time-for-parallelization and performance results of two parallel implementations of the same application (i.e. using Star-P and MPI), create a cost-benefit relationship from which others can decide if the loss is performance (if any) is worth the savings in development time.

## Using Star-P to Aid Hyperspectral Imaging

The first application that we chose for parallelization using Star-P is the Hyperspectral Image Analysis Toolkit (HIAT). The HIAT is a collection of image processing methods for the analysis of hyperspectral and multispectral images that was developed by the CenSSIS group at the University of Puerto Rico at Mayaguez.

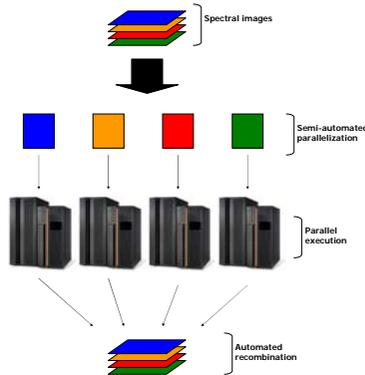


Figure 2: Star-P's approach to semi-automated parallelization

## Ease and Speed of Implementation

My first experience using Star-P involved integrating the entire HIAT (~20 functions) to run with Star-P, and took **30 hours**. The learning curve for using the software is virtually non-existent except for a few situations.

Not all of the functions benefited greatly from speedup since their execution time was already very short, but certain sections of code had to be rewritten to avoid performance degradation. Degradation in performance is caused by excessive communication between processors. Ways to avoid excessive communication are:

- Vectorizing code whenever possible
- Removing loops where conditions must be checked
- Being conscious of the distribution of data

## Ease and Speed of Implementation continued

Figure 3 displays the approximate time spent on different aspects of developing the parallel HIAT using Star-P.

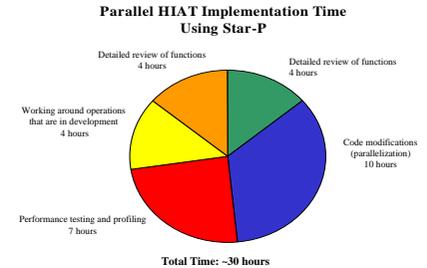


Figure 3: Summary of development effort for parallelizing HIAT using Star-P

## Speedup Results

The speed up results shown here come from an execution on the Teranode system, which contains two dual-core processors (for a total of 4, 3 of which participated in executing the algorithms) and has 18GB of RAM. The image being operated on was approximately a 260MB file, with pixels of double data type.

## Execution Time of HIAT Functions

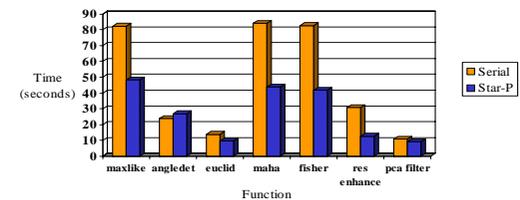


Figure 4: Performance comparison of serial versus Star-P parallelized HIAT functions

The results present the execution time for one iteration of each function. Based on user options, many iterations of each function may be performed. Also, increasing the image size causes the Star-P execution time to increase less quickly than the serial version.

## Task Parallelizing Tomosynthesis Reconstruction

Tomosynthesis reconstruction creates a 3D image from multiple 2D x-rays with the goal of visualizing any tumors that may be in the body. The serial version of the code takes longer than is practical, so a parallel MPI version was also created. However, the process of parallelizing the code using MPI was time consuming, and we are going to try to achieve the same results in much less time using Star-P.

A functioning parallelized version of Tomosynthesis reconstruction was created using Star-P in about 10 hours (with some extra time with development issues). This is very useful for creating parallel code with C that can be used from a Matlab interface.

