



### ■ Easy programming and interactive parallel computing for Science and Technology

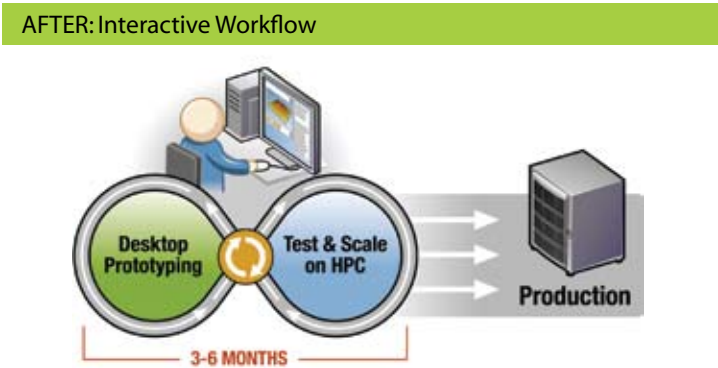
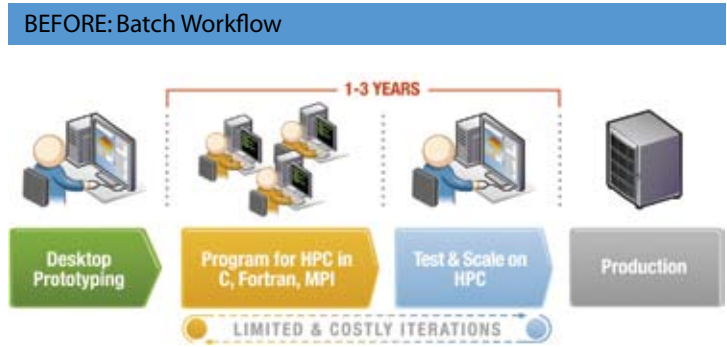
- Accelerate your time-to-insight
- Get the most from your experience and existing code
- Maximize the value of your multi-core servers and clusters

### ■ The interactive desktop tool you want, the supercomputing power you need

- User-friendly programming and computing paradigm
- Boosting productivity of teams skilled in MATLAB

The scale and pace of the bulk of scientific and technical computing tasks has outgrown the desktop limits while the programming of supercomputers has not become simple or inexpensive. Star-P enables easy parallel computing on the low-cost, multi-core servers and clusters and dramatically extends the capabilities of traditional desktop-based technical computing environments.

Productivity boost brought by Star-P to the teams of MATLAB-skilled domain experts is matched by the economic benefits of getting the most out of the computing power of next generation servers and clusters. Star-P enables interactive workflow for large-scale scientific and engineering computing, eliminating the need for intermediate steps of reprogramming the code in C, Fortran, and MPI, and dramatically shortening the time to insight.



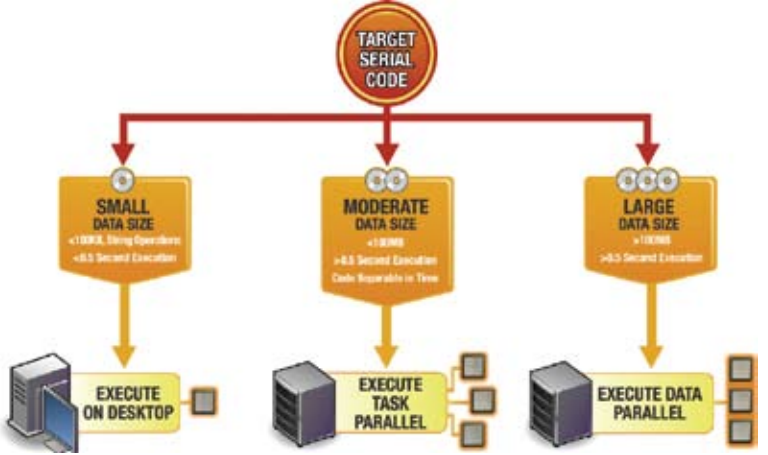
### Simple and Efficient Parallel Computing for Science and Technology

The programmer interacts with standard MATLAB desktop environment, enhanced with few simple Star-P commands

- Tedious parallel programming is eliminated (No MPI, C, Fortran)
- Conversion from serial to parallel is easy and mostly automated

Star-P with MATLAB environment supports

- Serial computing (plain MATLAB)
- Data-Parallel computing (\*p tag)
- Task-Parallel computing (ppeval command)
- Additional functions for data- and task-parallel computing through Star-P Connect™ Library API link



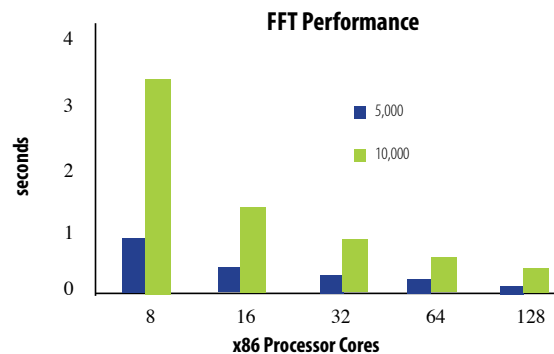
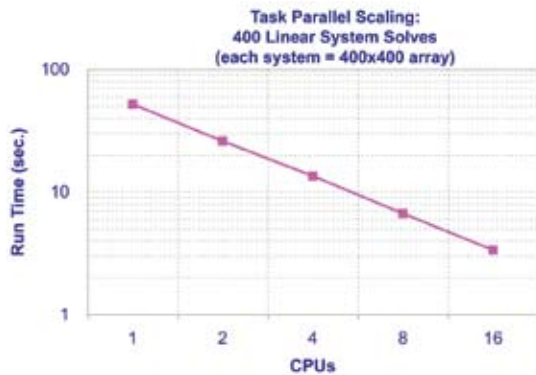
## Functions Available with Star-P for Parallel Computing

- 500+ MATLAB functions and operators
- 250+ MATLAB toolbox functions (statistics, signal processing, optimization, ...)
- 600+ library functions
- Unlimited user or community functions via Star-P Connect
- Star-P specific functions: ~20 programming or environment control and support functions
- For the most up-to-date information, visit the <http://www.interactivesupercomputing.com/products/starpanmatlab.php>

## Scale and Performance

Star-P has been tested in a variety of environments, and has shown a clear performance scaling with the number of processors involved in parallel computation. Additionally, Star-P uses all memory in a multi-core server or cluster for data processing, enabling interactive simulations with very large, real-life data sets.

The scale and performance of Star-P computing depends on the properties of the hardware platform and cluster interconnects. Best performance is achieved when the choice of computing mode is best matched to all segments of the algorithm being tested or simulated. Star-P has been tested and scaled to manipulate 3 Terabyte-sized matrices, on machines reaching 512 processors. Star-P runs on servers with x86-64 multi-core microprocessors such as Intel's Xeon and AMD's Opteron, and Intel's multi-core Itanium. Supported client operating systems include Windows XP and Vista, and SUSE and Redhat Linux. Supported server operating systems include SUSE and Redhat Linux.



## Data Parallel and Task Parallel Computing

Leveraging both data- and task-parallel computing is necessary in many scientific and technical simulations. Star-P enables users to work in both modes and to seamlessly interoperate between the two.

Star-P's data-parallel mode enables algorithms requiring large-scale memory access and inter-processor communication, often called "global array computing", such as those found in matrix manipulation and signal processing applications. Star-P's task-parallel mode is ideally suited for parallelization of algorithms often called "embarrassingly parallel," where computations can be naturally broken up into independent processes such as Monte Carlo simulation, or parallelization of For loops.

### Task Parallel Example

```
%Generate the Fourier Transform on 10 degree spacing
angles = linspace(0,360,37);
%Serial Version
load('brain.mat','A');
for i = 1:length(angles);
    FFTangles(:, :, i) = genFFTangles(angles(i), A);
end
%Parallel Version
ppload('brain.mat','A');
FFTangles = ppeval('genFFTangles', split(angles), bcast(A));
```

With Star-P, task-parallel computing is triggered by using the "ppeval" function call, analogous to MATLAB's feval (or "function evaluate") command. In this example, the use of "ppeval" replaces the For loop used in the serial version of the code, to carry out in parallel FFT operations on a brain image at 10 different rotations.

### Data Parallel Example

```
% explicitly parallel with *p
n=10000*p
% implicitly parallel
A = rand(n, n);
% implicitly parallel
x = randn(n, 1);
% implicitly parallel
y = zeros(size(x));
while norm(x-y) / norm(x) > 1e-11
    y = x;
    x = A*x;
    x = x / norm(x);
end;
```

Simple MATLAB script for finding the eigenvector of a random matrix. Adding the \*p construct makes variables parallel. Through propagation, related variables also become parallel. Functions on parallel variables are transparently "overloaded" and processed on a multi-core server or cluster.

INTERACTIVE  
supercomputing

135 Beaver Street, Waltham, MA 02452 • Phone: +1.781.419.5050 • Fax: +1.781.419.6050 • [www.interactivesupercomputing.com](http://www.interactivesupercomputing.com)