



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

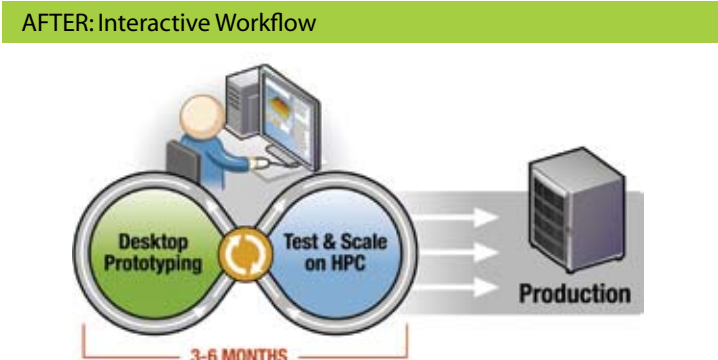
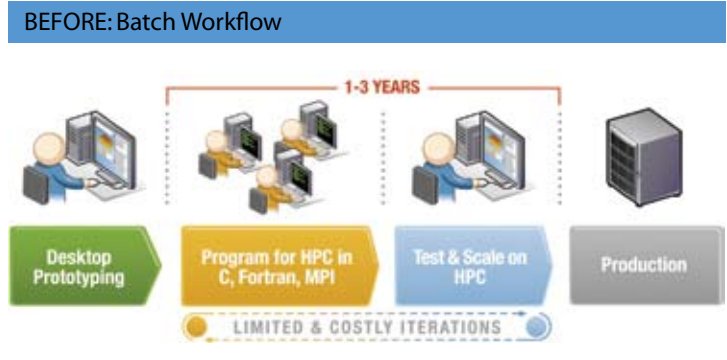
The interactive desktop tool you want, the supercomputing power you need

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

Star-P enables R users to access many powerful parallel programming idioms right from their familiar environment, to address large and complex statistical computations.

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.

Productivity boost brought by Star-P to the teams of R-skilled domain experts is matched by the economic benefits of getting the most out of the computing power of next generation servers and clusters.



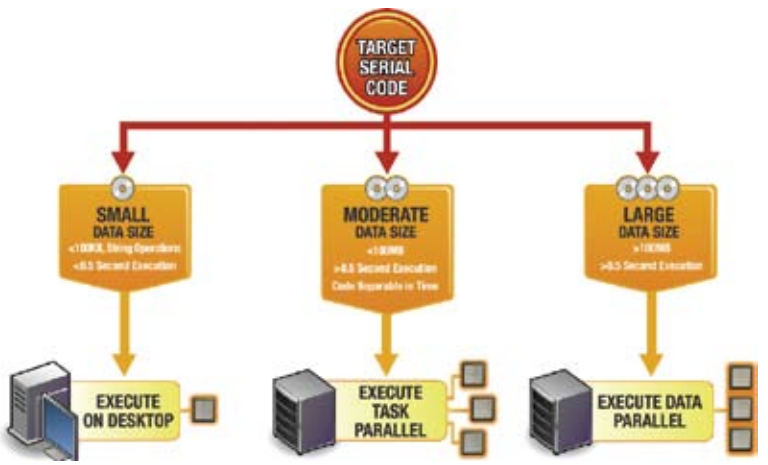
Simple and Efficient Parallel Computing for Science and Technology

The programmer interacts with standard R 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 R environment supports

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



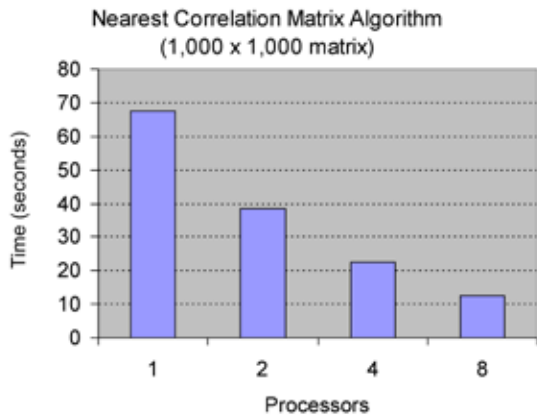
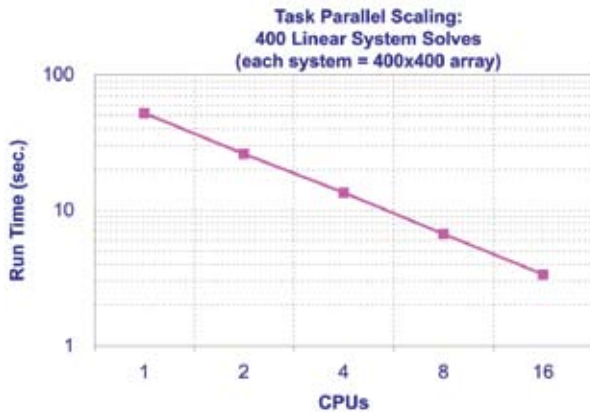
Functions Available with Star-P for Parallel Computing

- Task-parallel computing: 500+ R functions and operators
- Data-parallel computing: over 80 of the most popular (“blockbuster”) computing functions
- Unlimited user or community functions via Star-P Connect Library API link
- Star-P specific functions: ~20 programming or environment control and support functions

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 statistics 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.

Example

```
# Nearest correlation matrix algorithm
# Higham, NJ. 2002. Computing the nearest correlation matrix --
# a problem from finance. IMA J. Numerical Analysis 22:329-343.
```

```
M <- ppback(M)           # move data to HPC server
U <- array(0,dim(M))
Y <- M; X <- M
```

```
iter <- 0
while (iter < maxiter) {
  # Remember old values to check for convergence
  Yold <- Y; Xold <- X
```

```
  # Solve for eigenvalues and eigenvectors
  T <- Y-U
  ES <- eigen(sqrtW %*% T %*% sqrtW, TRUE)
  ESindex <- ES$values > 1e-4*max(ES$values)
  QQ <- ES$vectors[,ESindex]
```

```
  # Update solution
  X <- QQ %*% diag(ES$values[ESindex]) %*% t(QQ)
```

```
  U <- X - T
  Y <- X
  diag(Y) <- 1
```

```
  iter <- iter + 1
  # Check for convergence
  if (max( InfNorm(X-Xold)/InfNorm(X),
          InfNorm(Y-Yold)/InfNorm(Y),
          InfNorm(Y-X)/InfNorm(Y) ) <= 1e-5 ) {
    break
  }
}
```

An R script to compute the nearest correlation matrix. The only addition required to parallelize the script is highlighted in blue. Once the data is moved to the server, the parallelism propagates automatically. Functions on parallel variables are transparently “overloaded,” and processed on the server.

Contact:

Headquarters - James River Technical, Inc., 4439 Cox Road, Glen Allen, VA 23060, tel: 804.935.0150, sales@jrtech.com, www.jrtech.com

INTERACTIVE
supercomputing

JAMES
RIVER
TECHNICAL, INC.

© Copyright 2004-2007, Interactive Supercomputing, Inc. and its licensors. All rights reserved.

Star-P® and the “star” logo are trademarks of Interactive Supercomputing. R® is a registered trademark of The MathWorks, Inc. ISC’s products are not sponsored or endorsed by The MathWorks, Inc. or by any other trademark owner referred to in this document.