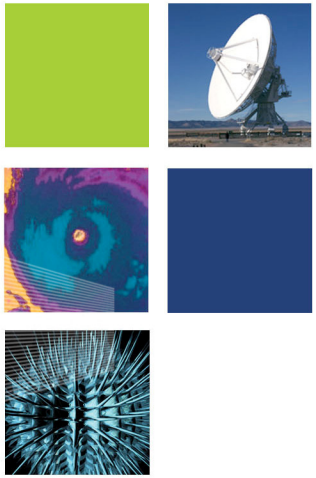
Star-P: Parallel Applications via a MATLAB® Environment

November 8, 2006



Eckart Jansen
Carlos Rojas
Ken Stanley

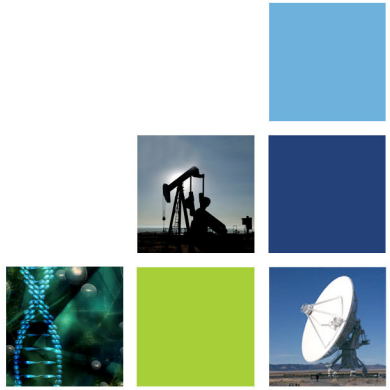
INTERACTIVE supercomputing

Agenda

- Star-P: Vision, platform, market traction
- How Star-P works
- Overloading “\” with Amesos solvers
- Stratimikos interface

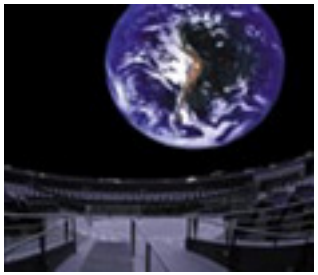
Trademark Usage Notice: STAR-P™ and the "star" logo are trademarks of Interactive Supercomputing, Inc. MATLAB® is a registered trademark of The MathWorks, Inc. Other product or brand names are trademarks or registered trademarks of their respective holders. ISC's products are not sponsored or endorsed by The Mathworks, Inc. or by any other trademark owner referred to in this document.

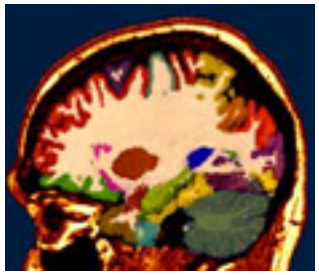
INTERACTIVE supercomputing



Our Vision

"Bridge the gap between easy-to-use desktop modeling, simulation and development tools on personal computers with the power, scalability and low cost of parallel computer systems, clusters and grids"

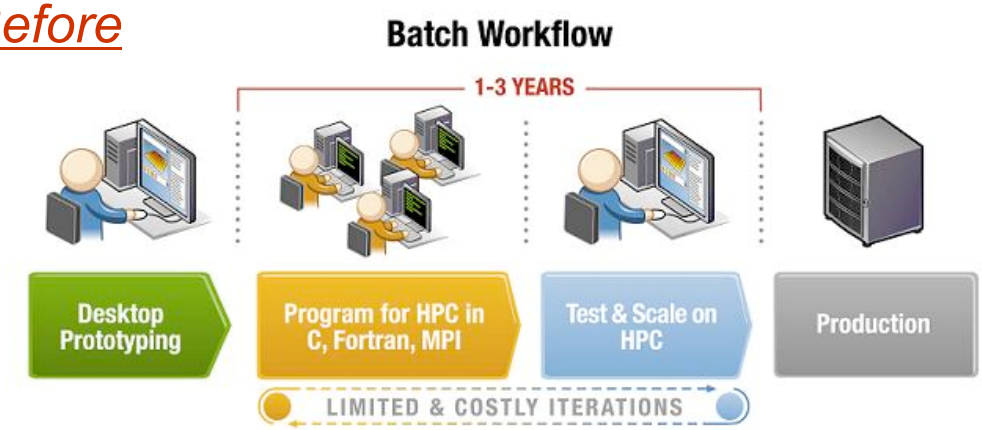




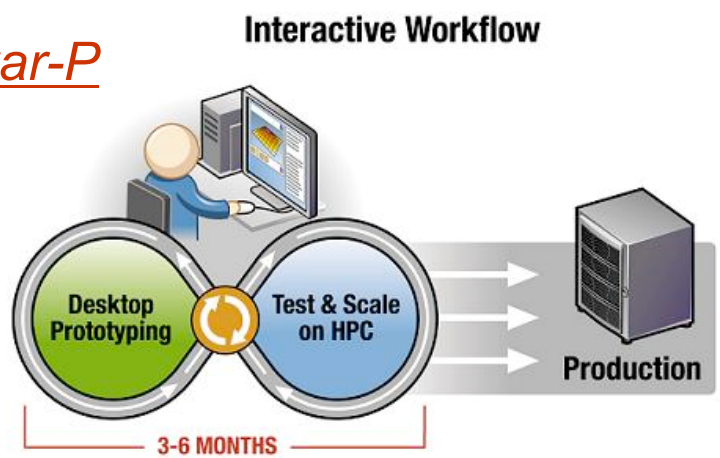
INTERACTIVE supercomputing

A Difference that Matters

Before



After, with Star-P



INTERACTIVE supercomputing

Star-P™ Enables:

Getting to the Top Objective

- Accelerating time-to-insight

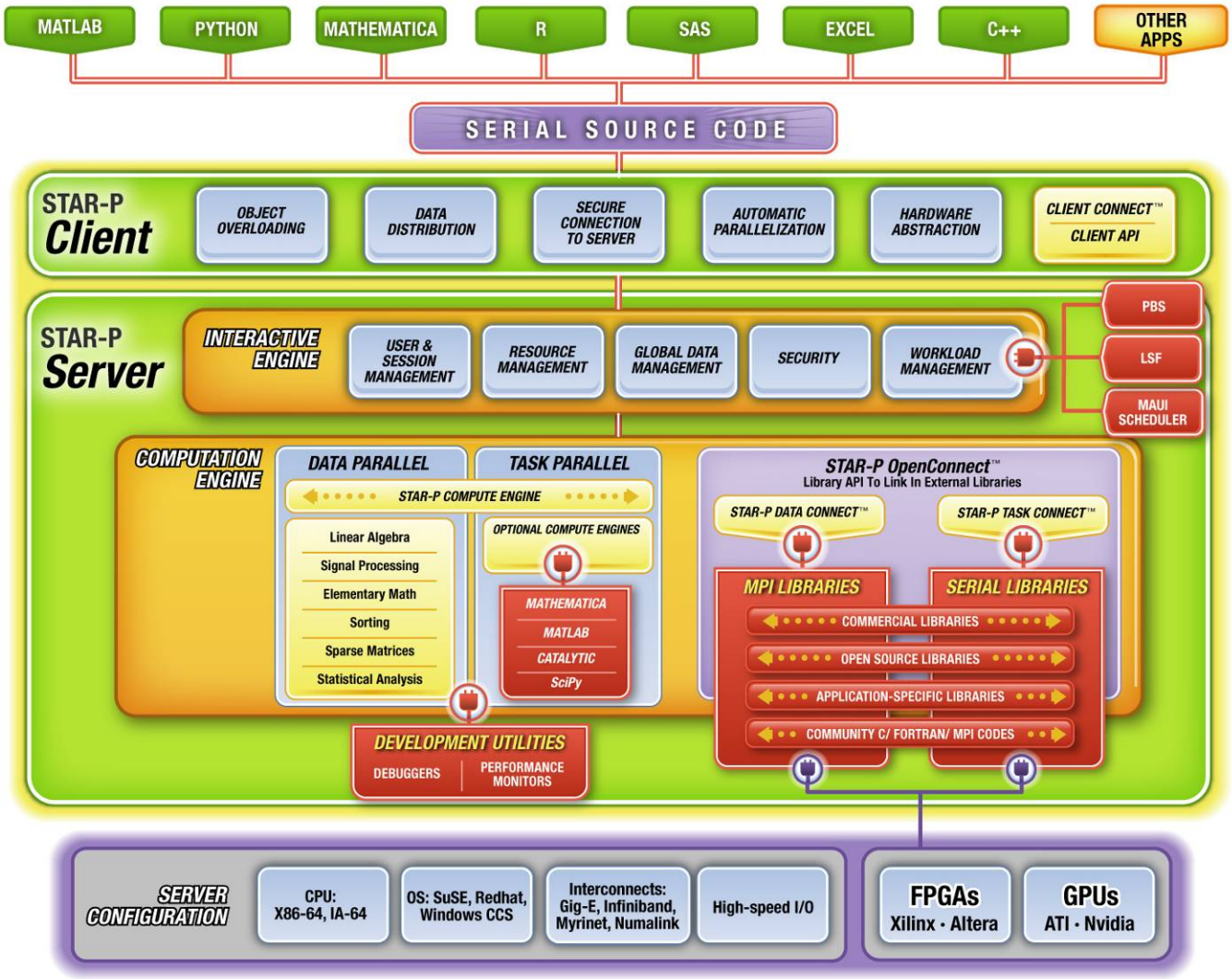
Efficiency

- Domain expert-friendly programming and computing paradigm
- Reuse of existing code from a broad range of sources

Economics

- Boosting productivity of HLL-skilled teams *(today's HLL: MATLAB®)*
- Getting the most out of the new, low-cost, multi-core servers and clusters
- Increased ROI, through the network effect of connecting multiple HLL environments with multiple sources of code, solvers, libraries

Star-P Platform



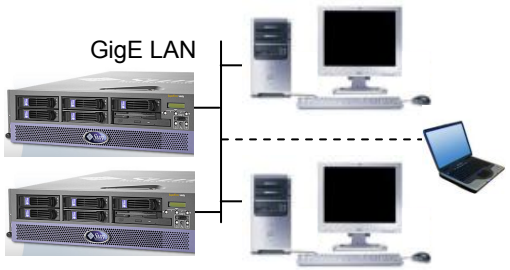
INTI supercomputing

Product Status

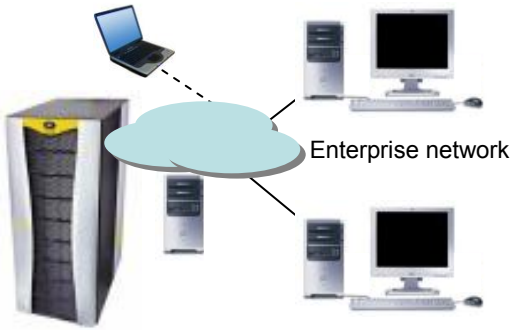
- **Market experience and feedback**
  - 10+ Altix sites, 10+ Opteron/Xeon sites, 100+ user codes
  - Success stories, application examples, and papers → *website*
- **Field tested configurations**
  - Server operating systems: Linux SUSE SLES or Redhat RHEL
  - Server hardware: SMP servers and clusters, x86/64, IA64
  - Client: Linux, Windows, x86, x86/64, MATLAB®
- **Release 2.**4 shipping
  - Compatible configurations → See http://www.interactivesupercomputing.com

Multi-core SMP Servers

Multi-core Clusters



Traditional HPC Servers

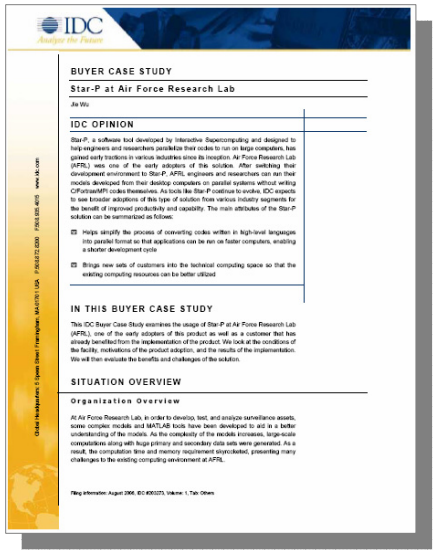


INTERACTIVE supercomputing

Market Traction

- Strong market interest
- Two dozen early adopters
- Building a channel
- Strong interest in open platform among users and partners

- Baylor College of Medicine
- Cleveland Clinic
- Dartmouth Medical School
- Duquesne University
- General Atomics
- M.D. Anderson Cancer Center
- Morgan Stanley
- National Cancer Institute
- National Center for Supercomputing Applications
- National Oceanic and Atmospheric Association
- Pennsylvania State University
- …



IDC

BUYER CASE STUDY

Star-P at Air Force Research Lab

Jie Wu

IDC OPINION

Star-P, a software tool developed by Interactive Supercomputing and designed to help engineers and researchers parallelize their codes to run on large computers, has gained early traitions in various industries since its inception. Air Force Research Lab (AFRL) was one of the early adopters of this solution. After switching their development environment to Star-P, AFRL engineers and researchers can run their models developed from their desktop computers on parallel systems without writing C/Fortran/MPI codes themselves. As tools like Star-P continue to evolve, IDC expects to see broader adoptions of this type of solution from various industry segments for the benefit of improved productivity and capability. The main attributes of the Star-P solution can be summarized as follows:

- Helps simplify the process of converting codes written in high-level languages into parallel format so that applications can be run on faster computers, enabling a shorter development cycle
- Brings new sets of customers into the technical computing space so that the existing computing resources can be better utilized

IN THIS BUYER CASE STUDY

This IDC Buyer Case Study examines the usage of Star-P at Air Force Research Lab (AFRL), one of the early adopters of this product as well as a customer that has already benefited from the implementation of the product. We look at the conditions of the facility, motivations of the product adoption, and the results of the implementation. We will then evaluate the benefits and challenges of the solution.

SITUATION OVERVIEW

Organization Overview

At Air Force Research Lab, in order to develop, test, and analyze surveillance assets, some complex models and MATLAB tools have been developed to aid in a better understanding of the models. As the complexity of the models increases, large-scale computations along with huge primary and secondary data sets were generated. As a result, the computation time and memory requirement skyrocketed, presenting many challenges to the existing computing environment at AFRL.

SGI - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.sgi.com/

sgi

Home | How to Buy | About SGI | Worldwide

Search

Products & Services | Industry Solutions | Support & Downloads | Developers

Our Focus:

High-Performance Computing

Storage

Resources for:

Press

Investors

Resellers

Careers

Food for Thought

Technology Solutions seminar series

SGI Resource Centers

Stay abreast of the latest trends and tools

SGI®at AFRL

Scalable Processing Power for Satellite Radar Imaging

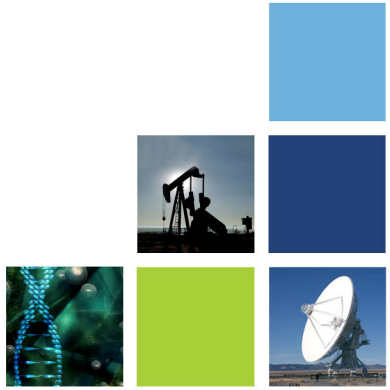
News & Highlights

- SGI Altix XE Cluster Solution Offers Simplicity, Advanced Software Environment
- SGI Plan of Reorganization Receives U.S. Bankruptcy Court Approval
- Louisiana Hosts One of World's Largest Supercomputers: a 4.1-terabyte SGI Altix 4700
- SGI and High-Performance Databases: Powering the Flexible Real-time Enterprise
- SGI Reorganization

more news | subscribe to our newsletter

Contact Us | Site Map | Trademarks | Privacy | Using this site means you accept its Terms of Use

INTERACTIVE supercomputing



How it works: Data Parallelism

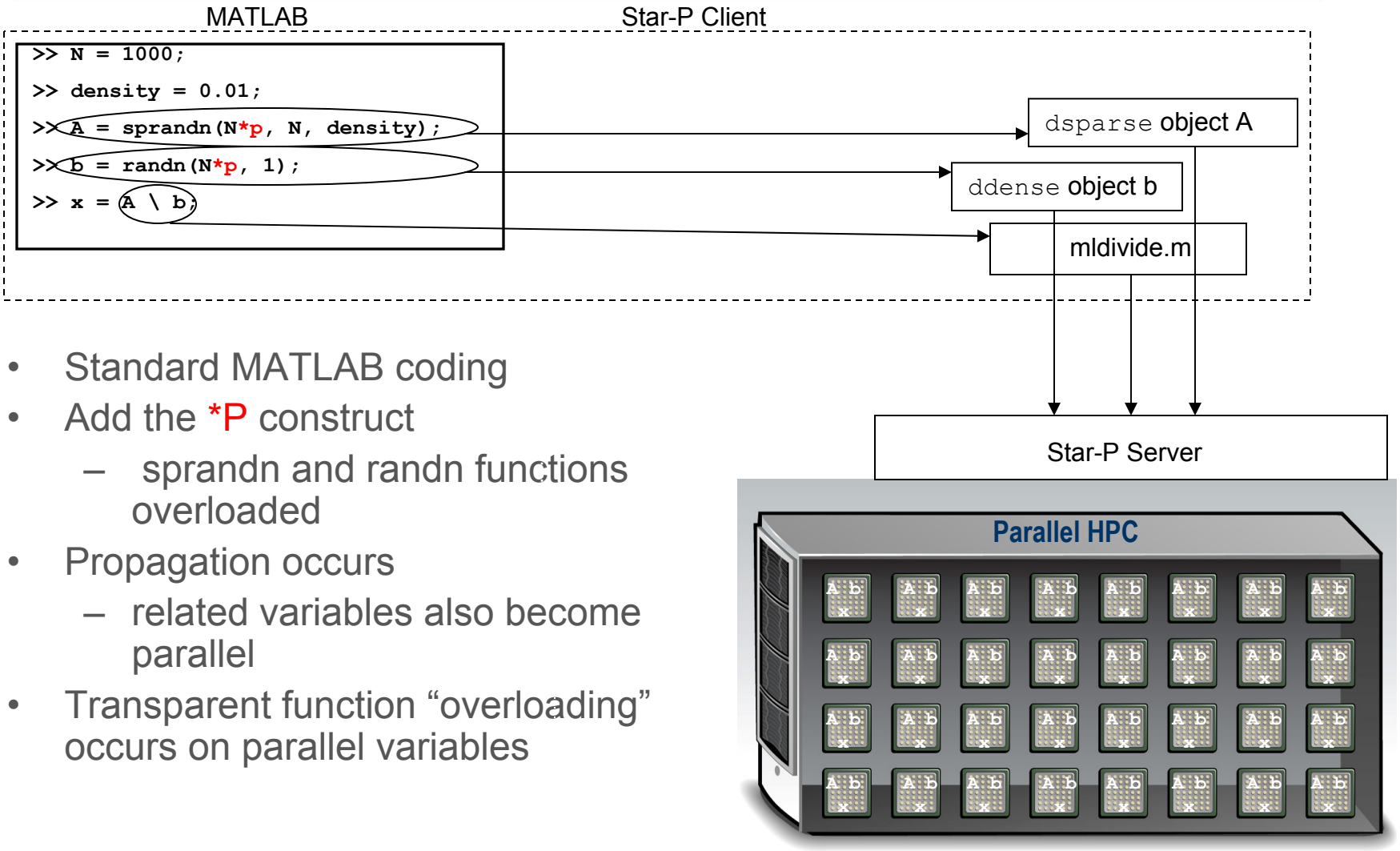
Fine-grained Problems:

- Standard MATLAB coding
- Add the *p construct
  - variables become parallel
- Propagation occurs
  - related variables also become parallel
- Transparent function "overloading" occurs on parallel variables

```
% 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

How it works



- Standard MATLAB coding
- Add the *P construct
  - sprandn and randn functions overloaded
- Propagation occurs
  - related variables also become parallel
- Transparent function "overloading" occurs on parallel variables

INTERACTIVE supercomputing

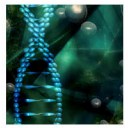
Amesos: Not just from C/F/MPI anymore

From MATLAB Prompt:

```
>> N = 1000;
>> density = 0.01;
>> A = sprandn(N*p, N, density);
>> b = randn(N*p, 1);
>> % First call the default sparse solver
>> x_default = A \ b;
>> % Now call KLU
>> global DsparseSolver; DsparseSolver = 'Amesos_KLU' ;
>> x_klu = A \ b;
>> % Compute the relative error
>> err = norm(x_default - x_klu)/norm(x_default) * 100
```

Goals:

- Develop tool for interactive application prototyping and development from a desktop High Level Language tool (MATLAB)
- Demonstrate basic interface to Amesos for access to parallel sparse direct solvers

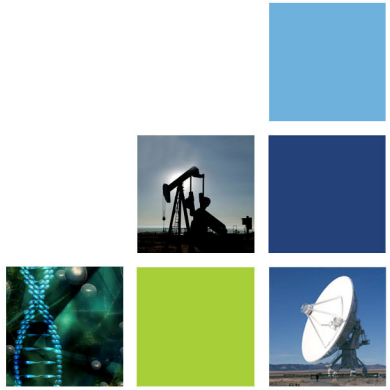


INTERACTIVE supercomputing

Implementation Overview

Two Main Steps

1. Server Side: Build the interface from Star-P to Amesos on the server and register the function with the server runtime system
2. Client Side: Load the server side interface on the client and map an appropriate `ddense`, `dsparse` or `ddensend` operator to the newly loaded function

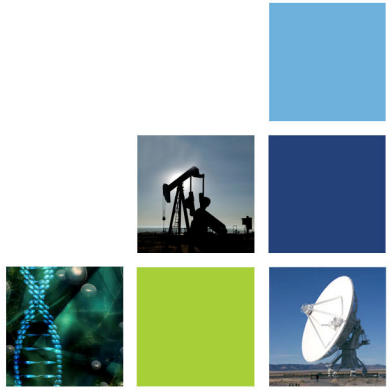


INTERACTIVE supercomputing

Server Side

The implementation on the server side involves

1. Write a function with a specified signature that validates the input arguments
2. Copy data from a `dsparse` left-hand side and a `ddense` right-hand side, passing it into a Trilinos data structure that defines a sparse linear system
3. Invoke the solver and return the solution back to client



INTERACTIVE supercomputing

Step 1: Function Definition

- Declare SDK function

```
extern "C"
void SDK_Amesos (PPServer &inServer, PPArgList &inArgs, PPArgList &outArgs)
```

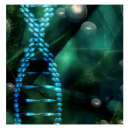
- Instantiate SDK object

```
StarpSDK starpSDK (inServer, inArgs, outArgs);
```

- Validate inputs

```
ThrowIfFalse (starpSDK.check_input_arg_count (2), "SDK_Amesos requires exactly two arguments.\n");
```

```
ntuple_t global_sizes_A = matrix_A->layout_desc().global_size_array();

int global_rows_in_A = static_cast<int> ( global_sizes_A[0]);
int columns_in_A     = static_cast<int> ( global_sizes_A[1]);
ThrowIfFalse (global_rows_in_A == columns_in_A,
      "The first argument to SDK_Amesos must be a square matrix.\n");
```



INTERACTIVE supercomputing

Step 2: Define System to Solve

- Create an `Epetra Map` object

```
int             num globals       = -1;
const int       starting_index    = 0 ;
MPI_Comm mpi_   comm  = starpSDK.communicator() ;
Epetra_MpiComm epetra_comm (mpi_comm);

Epetra_Map epetra_map (num_globals, sparse_data->m_loc,starting_index, epetra_comm);
```

- Create a compressed row sparse matrix

```
bool use_statically_allocated_storage = true;
Epetra_CrsMatrix epetra_matrix(Copy, epetra_map, &entries_per_row[0],
                               use_statically_allocated_storage);

vector<int> int_column_index (max_num_entries);

for (int i = 0; i < local_rows ; i++) {
    for (int j = 0; j < entries_per_row[i]; j++) {
        int_column_index[j] = static_cast<int> ( idx_column_index[row_pointers[i] + j] );
    }

    epetra_matrix.InsertGlobalValues (sparse_data->fst_row+i, entries_per_row[i],
                                      &values[row_pointers[i]], &int_column_index[ 0 ]);
}

epetra_matrix.FillComplete() ;
```

INTERACTIVE supercomputing

Step 2: Define System to Solve cont.

- Create right-hand side and space for the solution

```
double *data_b = (double *) matrix_B->local_dense_data<ELTYPE>() ;
Epetra_MultiVector epetra_vector_B (Copy, epetra_map, data_b, local_rows_in_B, cols_in_B);
Epetra_MultiVector epetra_vector_X (epetra_vector_B);
```

- Create Amesos data structure, an `Epetra_LinearProblem` representing the sparse system of equations to solve and choosing the solver (KLU)

```
Epetra_LinearProblem epetra_linearproblem(&epetra_matrix, &epetra_vector_X, &epetra_vector_B);
Amesos Factory;
Amesos_BaseSolver* amesos_solver = Factory.Create("Amesos_Klu", epetra_linearproblem);
```

Step 3: Solve and Return Solution

- Symbolic factorization: Analyze sparsity structure, reorder elements to reduce non-0s, allocate storage

```
if ((ret = amesos_solver->SymbolicFactorization()) != 0) {
  os << "Failure during Amesos::SymbolicFactorization: error code "
     << ret << "\n" ;
  throw os.str() ;
}
```

- Numerical factorization: Perform forward Gaussian elimination and compute entries in L and U factors

```
if ((ret = amesos_solver->NumericFactorization()) != 0) {
  os << "Failure during Amesos::NumericFactorization: error code "
     << ret << "\n" ;
  throw os.str() ;
}
```

- Back substitution

```
if ((ret = amesos_solver->Solve()) != 0) {
  os << "Failure during Amesos::Solve: error code " << ret << "\n";
  throw os.str() ;
}
```

Step 3: Solve and Return Solution cont.

- Obtain Star-P `matrix_factory`, create `ddense` object, initialize it using the Epetra object's solution

```
matrix_factory_t MatFactory = starpSDK.get_matrix_factory ();
matrix_ptr matrix_X = MatFactory.clone (matrix_B);

double *sdk_data_x = (double*) matrix_X->local_dense_data<ELTYPE> ();
double *epetra_data_x ;
int lda_x ;
epetra_vector_X.ExtractView(&epetra_data_x, &lda_x);

int local size B = lda x * cols in B ;

for (idx_t i = 0; i < local_size_B ; ++i) {
  sdk_data_x[i] = epetra_data_x[i] ;
}
```

- Send the Star-P `ddense` object back to the client

```
starpSDK.add_output_matrix (matrix_X);
```

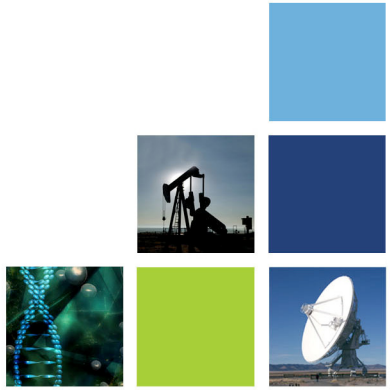
- Register user-defined function to name "`SDK_Amesos`"

```
extern "C"
int starpInit (PPPackage *pack)
{
  StarpSDK::register_func(pack, "SDK_Amesos", &SDK_Amesos);

  return 0;
}
```

Client Side

The implementation on the client side involves

1. Loading the solver into the Star-P client at runtime
2. Implementing the dispatcher function
3. Overloading the “\”operator for sparse matrices



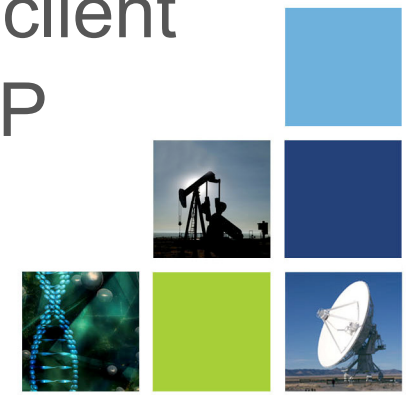
INTERACTIVE supercomputing

Client Step 1: Load the Solver

LoadAmesos.m

```
%
%  SDK_Amesos_path should point to the shared library file built on the server
%

SDK_Amesos_path ='/usr/local/starp/sdk/examples/amesos/server/SDK_Amesos.so';
global MyPath;

%
%  MyPath should point to the directory that this file resides in on client
%

MyPath ='/usr/local/starp/sdk/examples/amesos/client';

ppinvokeloadpackage   (SDK_Amesos_path);
```

- Set path/file on the server containing the SDK shared object
- Set path to overloading functions on client
- Load the shared object into the Star-P server



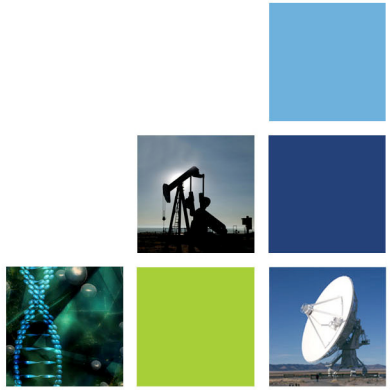
INTERACTIVE supercomputing

Client Step 2: Dispatch

SDK_Amesos

```
function x = SDK_Amesos( A, b )

    x = ppinvoke ('SDK_Amesos', A, b );
```

- Invocation is done using `ppinvoke` command that is passed the name of the SDK function and all the arguments expected by the function
- The function '`SDK_Amesos`' above calls the solver function and passes in the matrix and the right-hand side and stores the resulting `ddense` answer in the output variable



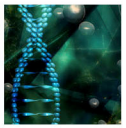
INTERACTIVE supercomputing

Client Step 3: Overload "\" Operator

MyPath/@ddense/mldivide.m

```
function x = mldivide(A,b)

    global DsparseSolver
    global MyDsparsePath

    if ~isempty( DsparseSolver ) && ~(strcmp(DsparseSolver,'Amesos_KLU') || ...
         (strcmp(DsparseSolver,'superlu') )
         fprintf('Sparse Solver %s not supported, reverting to the default method\n', ...
                  DsparseSolver)
    end

    if ( strcmp(DsparseSolver,'Amesos_KLU') && isreal(A) )
         x = SDK_Amesos( A, b ) ;
    else
         rmpath( MyDsparsePath )
         x = mldivide(A,b) ;
         path( MyDsparsePath, path )
    end
```

- Check global variable `DsparseSolver` for `Amesos_KLU`
- If set, call custom solver,
  else, revert back to default sparse solver



INTERACTIVE supercomputing

Testing the sparse solver

From MATLAB prompt

```
>> N = 1000;
>> density = 0.01;
>> A = sprandn(N*p, N, density);
>> b = randn(N*p, 1);
>> % First call the default sparse solver
>> x_default = A \ b;
>> % Now call KLU
>> global DsparseSolver; DsparseSolver = 'Amesos_KLU' ;
>> x_klu = A \ b;
>> % Compute the relative error
>> err = norm(x_default - x_klu)/norm(x_default) * 100
```

Stratimikos Interface

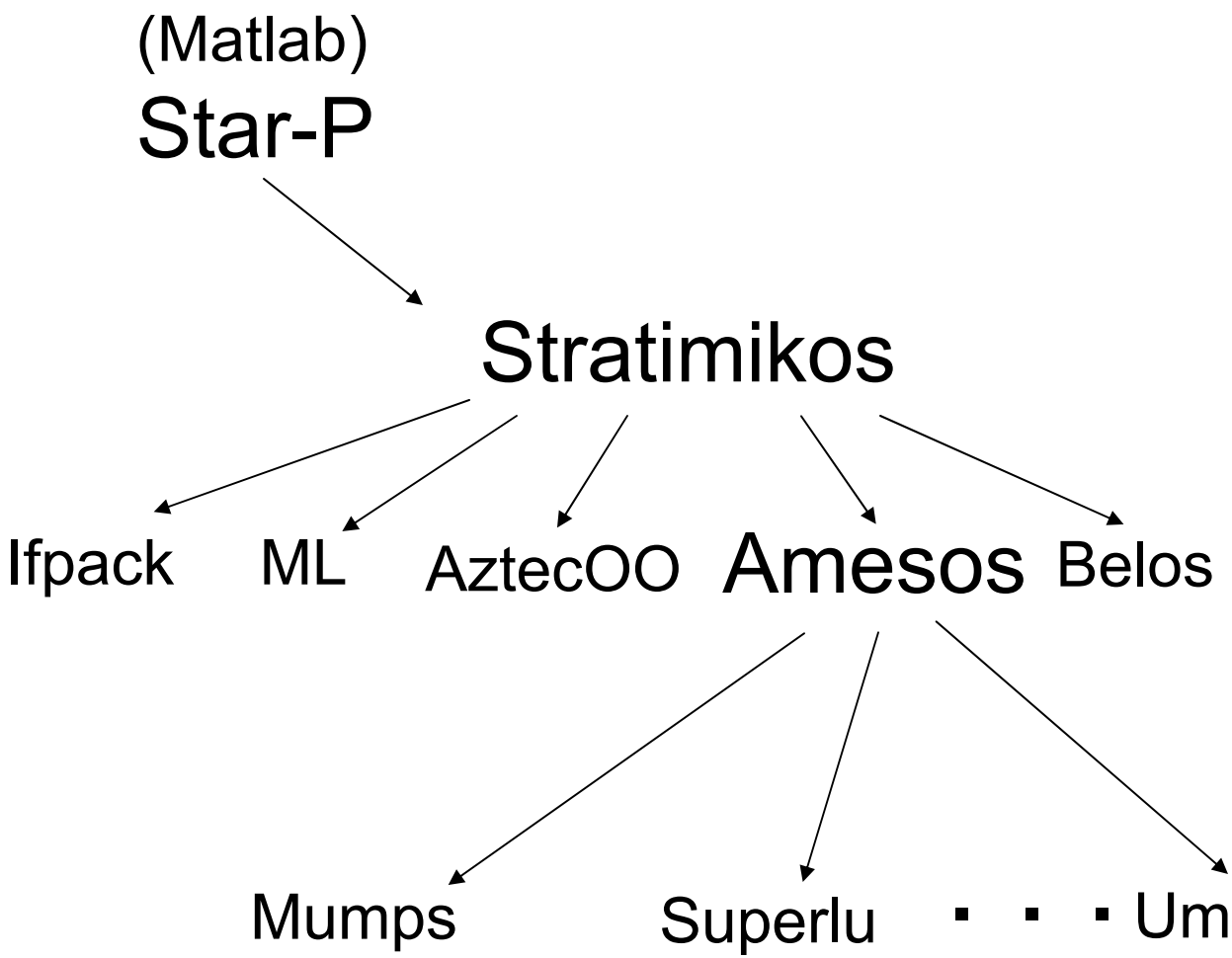
Goal: Make all Stratimikos supported solvers available through MATLAB

Benefits:

- Parameter list control
- Validation
- Real time documentation

INTERACTIVE supercomputing

Stratimikos supported solvers



INTERACTIVE supercomputing

Parameter list control

Proposed Star-P/Stratimikos/Mumps interface:

```
>>PL = { {'MatrixType', 'Symmetric'},
         { 'SymOpts',
           {{ 'Scaling', {'Diagonal'} }, ...
            { 'Pivoting', {'2by2 pivots'} } }

>>ppsetoption('SparseSolver','Mumps',PL);

>>x = A \ b ;
```

Parameter list is optional

INTERACTIVE supercomputing

Real time documentation

Display top level of the parameter list

```
>>ppgetoptions('SparseSolver','Mumps',PL);
>>ShowParamaterList(PL)
 'IterativeRefinement' [no]
 'MatrixType'->
 'SymmetricOptions' ->
 'NonSymmetricOptions' ->
        .
        .
        .
```

INTERACTIVE supercomputing

Real time documentation cont.

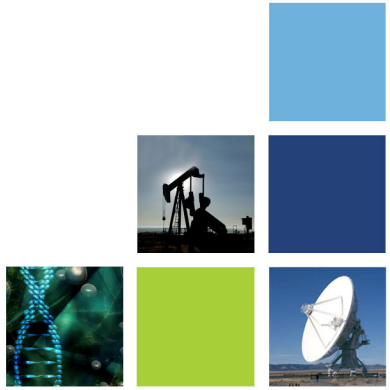
Display PL->'SymmetricOptions' + one level

```
>>ppgetoptions('SparseSolver','Mumps',PL);
>>ShowParamaterList(PL,'SymmetricOptions',2)
 'SymmetricOptions' ->
     'Scaling'->
          'NumericalFactorization'
          'NoScaling'
          'Diagonal'
          'AutomaticScaling' [default]
        .
        .
        .
```

INTERACTIVE supercomputing

Summary

- Star-P platform bridges the gap between familiar desktop tools and HPC
- Rapid prototyping of libraries and applications from MATLAB without compilation
- Access Amesos and other Trilinos solvers
- Stratimikos interface for parameter list control and real time documentation



INTERACTIVE supercomputing

Collaboration

- Have an application code you want to try with Star-P?
- Want to develop library or Trilinos package from Matlab?
- Interested in evaluating Star-P?

Carlos Rojas, crojas@interactivesupercomputing.com
Eckart Jansen, ejansen@interactivesupercomputing.com

http://www.interactivesupercomputing.com

INTERACTIVE supercomputing