The Lapack for Clusters (LFC) Project

Presented by

Piotr Luszczek

The MathWorks, Inc.

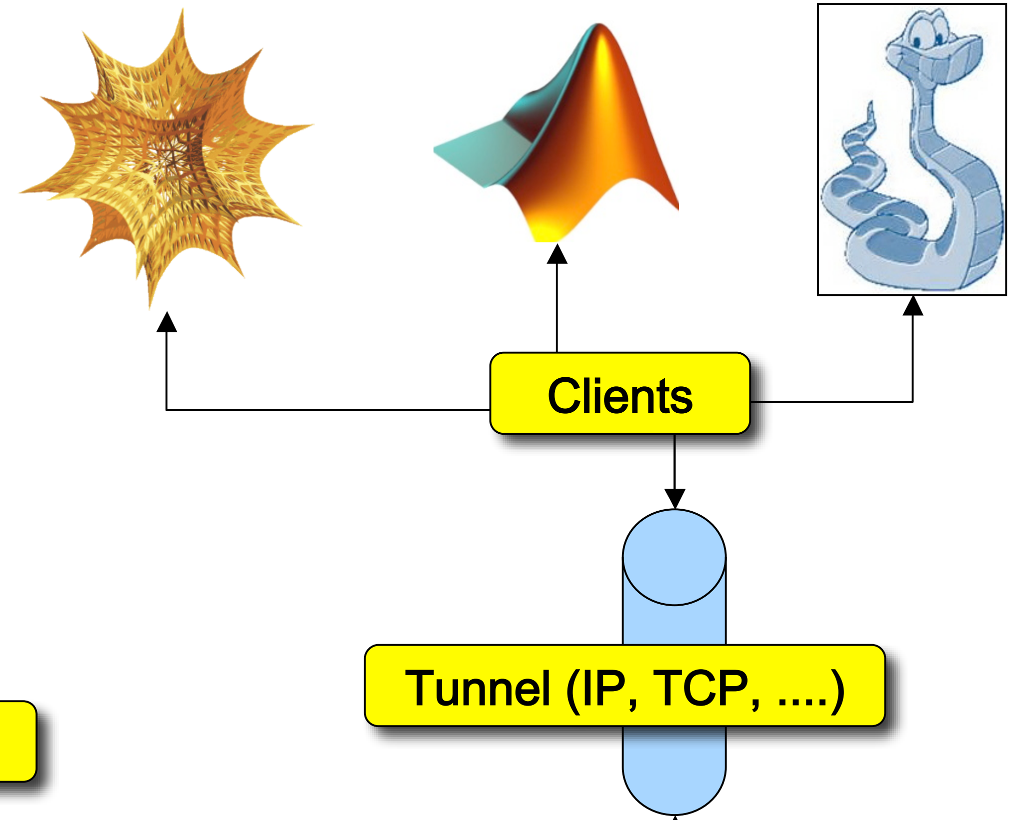
SC07

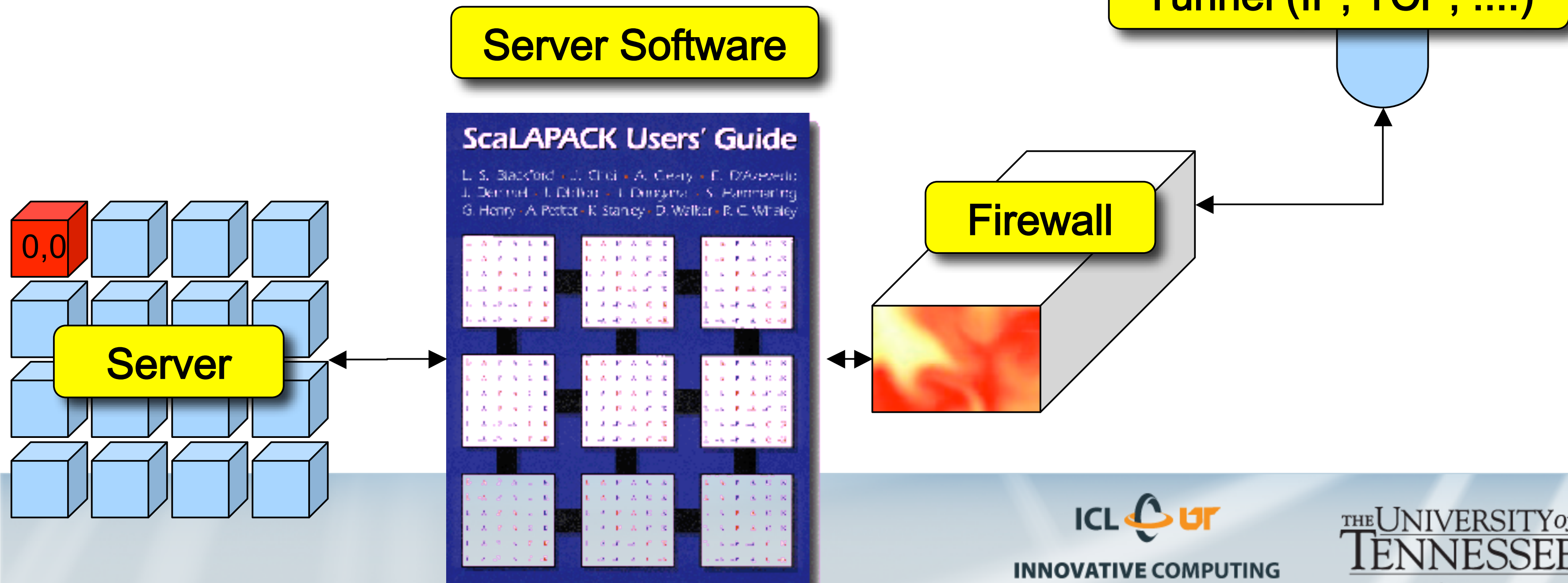
ICL UT INNOVATIVE COMPUTING LABORATORY

THE UNIVERSITY of TENNESSEE Computer Science Department

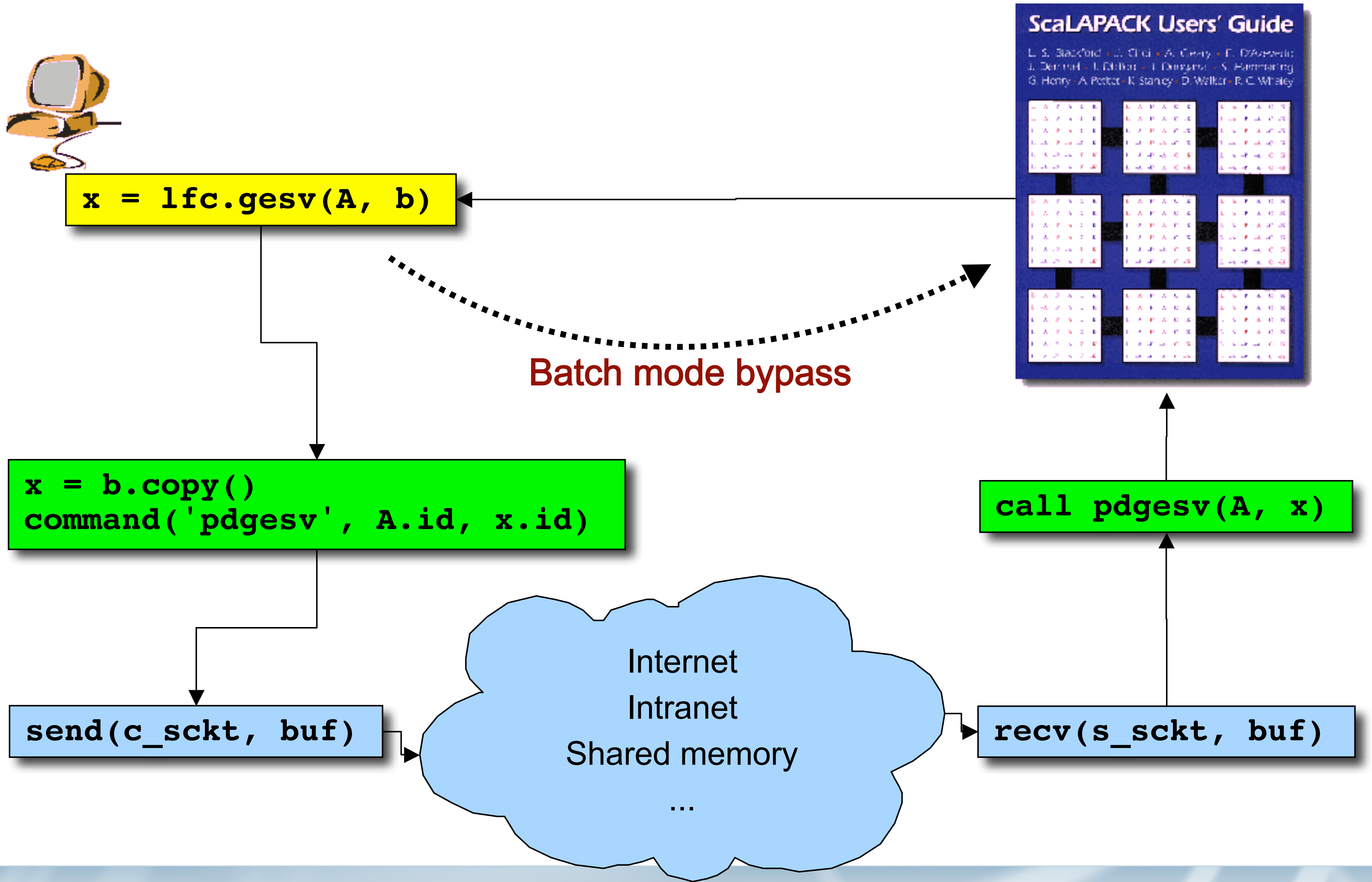
LFC: Overview

- Interactive clients
  - Mathematica
  - MATLAB
  - Python





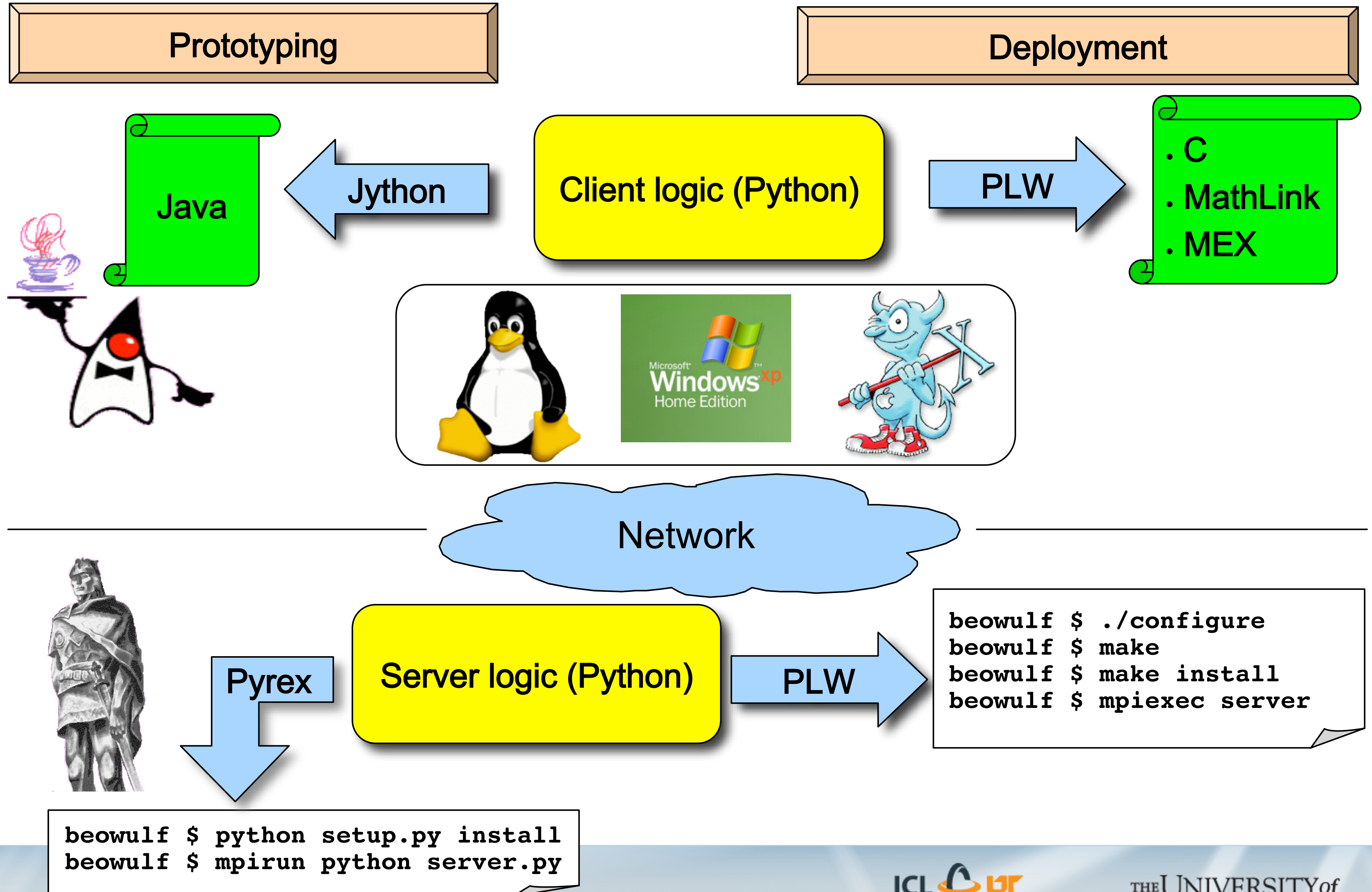
LFC: Behind the scenes



LFC: Current functionality

- Linear systems (via factorizations)
  - Cholesky: A = UTU   A = LLT
  - Gaussian: PA = LU
  - Least Squares: A = QR
- Singular- and eigen-value problems
  - A = UΣVT   (thin SVD)
  - AV=VΛ=AHV (symmetric AEP)
  - AV = VΛ   (non-symmetric AEP)
- Norms, condition-number estimates
- Precision, data types
  - Single, double
  - Real, complex
  - Mixed precision (by upcasting)
- User data routines
  - Loading/Saving
    - MPI I/O
    - ...
- Generating
  - Plug-ins
  - Moving
- More to come…
  - Now working on sparse matrices support

LFC: Design and implementation



LFC: Implemented MATLAB functionality

Name	Single	Double	S-Complex	D-Complex	Description
chol	√	√	√	√	Cholesky factorization: LL^T, U^TU
lu	√	√	√	√	LU factorization: PA=LU
qr	√	√	-1	-1	QR factorization: A = QR
qrp	-1	-1	-1	-1	QR factorization+pivoting: PA=QR
Svd	√	√	√	√	Singular-value decomposition: $A=U\Sigma V^T$
Eig	-1	-1	-1	-1	Eigenvalue problem: $AX=X\Lambda$
Syev	√	√	n/a	n/a	Symmetric eigenvalue problem
Heev	n/a	n/a	√	√	Hermitian eigenvalue problem
Diag	√	√	√	√	Diagonal matrix/vector
Trans	√	√	√	√	Matrix/vector transpose
Herm	n/a	n/a	√	√	Matrix/vector hermitian transpose
Norm	√	√	√	√	Matrix/vector norm
Cond	√	√	√	√	Condition number estimate
Inv	0	0	0	0	Explicit matrix inverse

ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE Computer Science Department

Sample LFC code: Linear system solve

- MATLAB with LFC (parallel):

```
n = lfc(1000);
nrhs = 1;
A = rand(n);
b = rand(n, 1);
x = A \ b;
r = A*x – b;
norm(r, ‘fro’)
```

- Python with LFC (parallel):

```
n = 1000
nrhs = 1
A = lfc.rand(n)
b = lfc.rand(n, 1)
x = lfc.solve(A, b)
r = A*x – b
print r.norm(‘F’)
```

- MATLAB – no LFC (sequential):

```
n = 1000;
nrhs = 1;
A = rand(n);
b = rand(n, 1);
x = A \ b;
r = A*x – b;
norm(r, ‘fro’)
```


ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE Computer Science Department

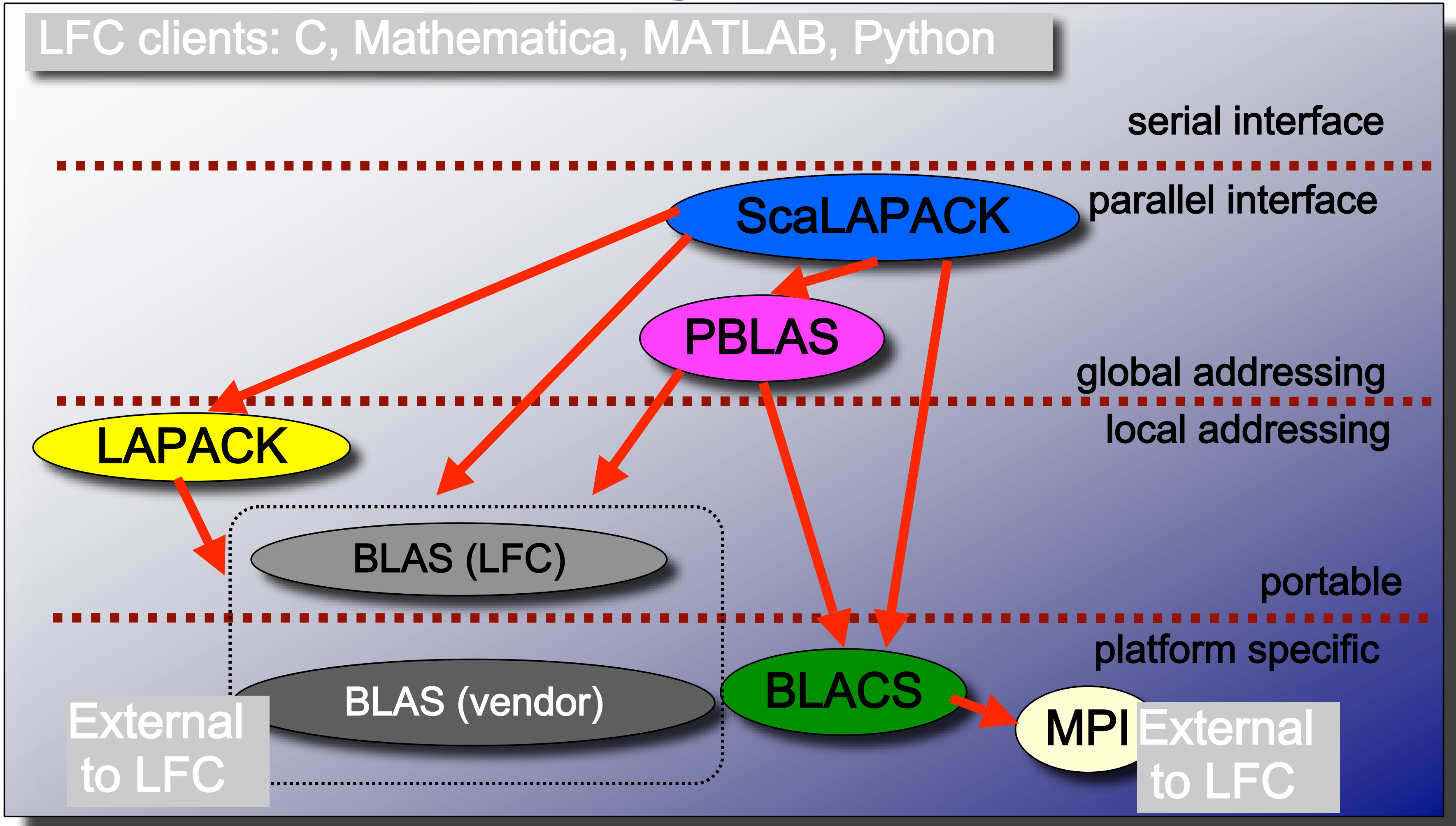
LFC's C interface: Sequential calls, parallel execution

- In-memory routines
  - `LFC_•gels()`
  - `LFC_•gesv()`
  - `LFC_•posv()`
- Limitations
  - Data must fit on caller
- Cluster state functions
  - `LFC_hosts_create()`
  - `LFC_hosts_free()`
  - `LFC_get_available_CPU()`
  - `LFC_get_free_memory()`

- Disk-based routines
  - `LFC_•gels_file_all()`
  - `LFC_•gesv_file_all()`
  - `LFC_•posv_file_all()`
- Limitations
  - Must pay I/O cost each time

ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE Computer Science Department

LFC's ease of deployment



- Only one file to download
- Just type: ./configure && make && make install

Software technology used by LFC

- Client
  - Python
    - Sockets
  - MATLAB
    - Embedded Java
    - Jython
      - Reuse of Python code
  - C/C++
    - Code:
      - fork()
      - execvp()
    - Shell (implicit)
      - mpirun
      - mpiexec
- Server
  - Libraries
    - MPI
    - BLAS
    - BLACS
    - LAPACK
    - ScaLAPACK
  - Languages
    - Python
    - Pyrex (C wrappers)
    - PLW
      - Python compiler
      - Translation to C

ICL UT INNOVATIVE COMPUTING LABORATORY
THE UNIVERSITY of TENNESSEE Computer Science Department

Contact

Piotr Luszczek

The MathWorks, Inc.
(508) 647-6767
luszczek@eecs.utk.edu

http://icl.cs.utk.edu/lfc/