Use of HPCC Software Libraries in Industrial Applications

Kenneth Hawick
David Walker

CRPC-TR95618
July 1995

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

Also available as SCCS-723 from the Northeast Parallel Architectures Center at Syracuse University.
Use of HPCC Software libraries in Industrial Applications

Kenneth A. Hawick
Northeast Parallel Architectures Center
Syracuse University, USA

&

David W. Walker
Mathematical Sciences Section
Computer Science and Mathematics Division
Oak Ridge National Laboratory

Presented at:
SPAA’95 — PPoPP’95 — UIWOPPS
Santa Barbara, California, July 17 – 22, 1995

NPAC Technical Report SCCS 723
Overview:

- Acknowledgments
- HPCC Libraries
- Industrial Driving Forces
- Pre-requisites for Building Libraries
- Case Study: ScaLAPACK Library
- Industrial Application of ScaLAPACK
- Conclusions
- Online Internet Resources
Acknowledgments:

- ScaLAPACK Development Team (Univ Tennessee / ORNL): Jaeyoung Choi; Susan Ostrouchov; Antoine Petitet; Clint Whaley; Jack J. Dongarra; David W. Walker.

- Computational Electromagnetics Project Application Team (NPAC/Syracuse Research Corp): Gang Cheng; Ken Hawick; Xianneng Shen; Jay Mortensen; Jim Lauer; Debra Wilkes.

- Industrial Numerical and Simulations Group at the Edinburgh Parallel Computing Centre (EPCC): Ken Hawick; Brian Wylie; Simon Chapple; Evan Welsh; Mark Sawyer; Julian Parker; Hon Yau.;

- CHIMP/Parallel Utility Library (PUL) Group at EPCC: Lyndon Clarke; Shari Trwein; Bob Fletcher; Alasdair Bruce; James Mills.
HPCC Libraries:

- encapsulate expertise
- can be extensively tested independently
- can provide portability across different vendor platforms
Industrial Driving Forces:

- Software development for HPCC platforms often more expensive in terms of development and testing than the hardware, for industrial reliability requirements.

- Investment only makes sense if software reuse - as libraries - is possible.

- Libraries preferable to template/skeletons as greater encapsulation allows better testing.

- although trade-off of performance against reliability and reuse exists - high performance still highly desirable!
Industrial Examples (UKMO):

- UK Meteorological Office: Unified Model is 150k lines of Fortran
- Parallel coding effort easier if higher than raw message passing libraries exist for grid manipulations.
- Multiple algorithm paradigms (data parallel for dynamics; task parallel for precipitation model; scattered spatial decomposition for data assimilation) requires interoperable library components with standard library interfaces.
- Parallel Utility Library (PUL) set designed and built at Edinburgh as a result.
**CHIMP/PUL Libraries:**

- CHIMP (Common High Level Interface to Message Passing), predated MPI and was attempt to provide a message passing that would allow partitioning of message tag space for building software libraries on top of.

- PUL (Parallel Utility Library) is a collection of libraries and skeletal templates built on top CHIMP, and now MPI.

- PUL examples include: general grid decomposition (like BLACS in ScaLAPACK); Task farm paradigm; scattered spatial decomposition; generalised blocked distributed file I/O. (Clarke et al, Edinburgh Parallel Computing Center)
Industrial Examples (RR):

- Rolls Royce (Aerospace Engine Design)
- Turbofan Hypersonic CFD simulation code of circa 30k lines Fortran.
- Code required linear algebra library such as ScaLAPACK which was not then available in 1991.
- Prototype was built using customised solver, but not able to be introduced into production due to high degree of code maintainance that would have been required.
- Supported library module would have allowed use of parallel platform in production instead of vector machines only.
Industrial Examples (AEA):

- UK Atomic Energy Authority - nuclear reactor simulations codes

- Large codes, need to be very reliable, and require extensive recurrency testing of all software modules - test/verification suite often larger than simulation code itself.

- Software libraries allow testing effort to be reused, as well as design verification/validation against other codes using the same library or a different library if on a vector platform.

- Use of CHIMP message passing library and Parallel Utility library for block decompositions allowed introduction of parallel computing into an otherwise ‘vector’ environment.
Industrial Examples (BAe):

- British Aerospace - radar cross section analysis codes.
- customised codes using Occam and assembly language to exploit cheap parallel hardware. No reliable dense linear algebra library existed in 1990 for HPCC parallel platforms.
- ScaLAPACK would (now) allow improved portable implementation.
Pre-requisites for HPCC Libraries:

- library typically built on a reliable message passing system.
- message passing calls actually used must be reliable and widely available - either in a portable library or standard such as PVM, MPI or CHIMP, or in the proprietary package available on target platforms (eg Intel NX/2, IBM EUI,...)
- for ease of development of multiple library modules, message tag space needs to be sensibly partitioned - for example alphanumeric group tags plus numeric message ID allows each library module to restrict itself to its own tag-space and ensure non-interference of library modules.
- well defined purpose for library is important for user as well as software designer. (contrast with some proprietary libraries which are ad-hoc collection of software packages). Difficult to maintain with time, and hard for user to know what to expect.
Case Study: ScaLAPACK Library - Motivation

- On shared memory vector supercomputers large, optimized software libraries exist:
  - BLAS, EISPACK, LINPACK, LAPACK,…
  - NAG, IMSL, ESSL,…
- Little such software runs efficiently on current and emerging parallel architectures
  ⇒ “Software Gap”
- Development of high-quality, portable software libraries for concurrent computers as a key enabling technology essential to more widespread use of HPCC platforms by industry as well as by academia.
Case Study: ScaLAPACK Library - Objectives

- Goal:

  To develop a library of high-quality, portable software for performing linear algebra computations on NUMA supercomputers, specifically MIMD distributed memory concurrent computers.

- LAPACK has already done this for workstations and shared memory computers.

- ScaLAPACK extends the functionality of LAPACK to distributed memory machines.

  ScaLAPACK = Scalable LAPACK

  i.e., we want the performance/node to stay constant as the problem size scales with the number of nodes.
Case Study: ScaLAPACK Library - Basic Problems

- Basic problems addressed by ScaLAPACK include:
- Linear systems: $Ax = b$
- Least squares: $\min_x \|Ax - b\|_2$, $A = U\Sigma V^T$
- Eigenvalues and vectors: $Ax = \lambda x$, $Ax = \lambda Bx$

ScaLAPACK and LAPACK use block-partitioned algorithms, so algorithm is expressed in terms of matrix-matrix operations performed using Level 3 BLAS, which maximizes data reuse in upper levels of memory, and reduces frequency of data movement between:

- shared memory and cache for shared memory machines;
- processors for distributed memory machines.
Case Study: ScaLAPACK Library - Building Blocks

- Basic Linear Algebra Communication Subprograms (BLACS) for communicating parts of a matrix. May be optimized for hardware.
- Parallel BLAS (PBLAS). Level 1, 2 and 3 BLAS routines for distributed matrices and vectors.
- Sequential BLAS. May be optimized for hardware.
- Matrix transpose routines.
- Data distribution transformation routines for dynamically changing data distribution.
Case Study: ScaLAPACK Library - BLACS

- Basic Linear Algebra Communication Subprograms communicate parts of: rectangular matrices; trapezoidal matrices.
- Processes are laid out on a 2D logical mesh
- Processes are referenced by location in topology
- Blocking point-to-point communication
- Collective communication over row, column or all of topology
  - broadcast
  - some reduction routines
- No message tags
- BLACS context is compatible with MPI communicator
Case Study: ScaLAPACK Library - PBLAS

- PBLAS perform Level 1, 2, and 3 BLAS operations on distributed matrices
- Matrices are global objects
- Matrices have a block cyclic data distribution
- PBLAS are a subset of the BLAS, but
  - no banded and packed storage schemes
  - no vector rotation routines
- Same calling sequence as BLAS except for each distributed matrix we have
  - global indices of start of matrix
  - descriptor array
Case Study: ScaLAPACK Library - Key Ideas

- Use block-partitioned algorithms to maximize data reuse in upper levels of memory
  * reduce cache misses
  * reduce frequency of communication
- Use Parallel BLAS (PBLAS) as main computational building blocks.
- Use Basic Linear Algebra Communication Subprograms (BLACS) to perform communication
- Hide parallelism within the PBLAS
- Fine-tune performance by adjusting data layout parameters

**Important:** The PBLAS make use of the sequential BLAS for which assembly coded versions exist for many processors.
Case Study: ScaLAPACK Library - Data Decomposition

- We want a data decomposition scheme that:
  - is practical,
  - is general-purpose,
  - gives good load balance,
  - can reproduce all the most commonly-used data distributions.

⇒ Block-Cyclic Distribution

- Partition matrix into blocks of $r \times s$ elements.
- Can regard processors as being arranged as a 2-D mesh, or template.

$$(m, n) \mapsto ((p, q), (b, d), (i, j))$$
## Case Study: ScaLAPACK Library - Block Cyclic Example

<table>
<thead>
<tr>
<th>p,q</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.0</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.0</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.0</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
</tr>
<tr>
<td>2</td>
<td>2.0</td>
<td>2.1</td>
<td>2.2</td>
<td>2.3</td>
<td>2.0</td>
<td>2.1</td>
<td>2.2</td>
<td>2.3</td>
<td>2.0</td>
<td>2.1</td>
<td>2.2</td>
<td>2.3</td>
<td>2.0</td>
<td>2.1</td>
<td>2.2</td>
<td>2.3</td>
</tr>
<tr>
<td>3</td>
<td>3.0</td>
<td>3.1</td>
<td>3.2</td>
<td>3.3</td>
<td>3.0</td>
<td>3.1</td>
<td>3.2</td>
<td>3.3</td>
<td>3.0</td>
<td>3.1</td>
<td>3.2</td>
<td>3.3</td>
<td>3.0</td>
<td>3.1</td>
<td>3.2</td>
<td>3.3</td>
</tr>
<tr>
<td>4</td>
<td>4.0</td>
<td>4.1</td>
<td>4.2</td>
<td>4.3</td>
<td>4.0</td>
<td>4.1</td>
<td>4.2</td>
<td>4.3</td>
<td>4.0</td>
<td>4.1</td>
<td>4.2</td>
<td>4.3</td>
<td>4.0</td>
<td>4.1</td>
<td>4.2</td>
<td>4.3</td>
</tr>
<tr>
<td>5</td>
<td>5.0</td>
<td>5.1</td>
<td>5.2</td>
<td>5.3</td>
<td>5.0</td>
<td>5.1</td>
<td>5.2</td>
<td>5.3</td>
<td>5.0</td>
<td>5.1</td>
<td>5.2</td>
<td>5.3</td>
<td>5.0</td>
<td>5.1</td>
<td>5.2</td>
<td>5.3</td>
</tr>
<tr>
<td>6</td>
<td>6.0</td>
<td>6.1</td>
<td>6.2</td>
<td>6.3</td>
<td>6.0</td>
<td>6.1</td>
<td>6.2</td>
<td>6.3</td>
<td>6.0</td>
<td>6.1</td>
<td>6.2</td>
<td>6.3</td>
<td>6.0</td>
<td>6.1</td>
<td>6.2</td>
<td>6.3</td>
</tr>
<tr>
<td>7</td>
<td>7.0</td>
<td>7.1</td>
<td>7.2</td>
<td>7.3</td>
<td>7.0</td>
<td>7.1</td>
<td>7.2</td>
<td>7.3</td>
<td>7.0</td>
<td>7.1</td>
<td>7.2</td>
<td>7.3</td>
<td>7.0</td>
<td>7.1</td>
<td>7.2</td>
<td>7.3</td>
</tr>
<tr>
<td>8</td>
<td>8.0</td>
<td>8.1</td>
<td>8.2</td>
<td>8.3</td>
<td>8.0</td>
<td>8.1</td>
<td>8.2</td>
<td>8.3</td>
<td>8.0</td>
<td>8.1</td>
<td>8.2</td>
<td>8.3</td>
<td>8.0</td>
<td>8.1</td>
<td>8.2</td>
<td>8.3</td>
</tr>
<tr>
<td>10</td>
<td>10.0</td>
<td>10.1</td>
<td>10.2</td>
<td>10.3</td>
<td>10.0</td>
<td>10.1</td>
<td>10.2</td>
<td>10.3</td>
<td>10.0</td>
<td>10.1</td>
<td>10.2</td>
<td>10.3</td>
<td>10.0</td>
<td>10.1</td>
<td>10.2</td>
<td>10.3</td>
</tr>
<tr>
<td>11</td>
<td>11.0</td>
<td>11.1</td>
<td>11.2</td>
<td>11.3</td>
<td>11.0</td>
<td>11.1</td>
<td>11.2</td>
<td>11.3</td>
<td>11.0</td>
<td>11.1</td>
<td>11.2</td>
<td>11.3</td>
<td>11.0</td>
<td>11.1</td>
<td>11.2</td>
<td>11.3</td>
</tr>
<tr>
<td>12</td>
<td>12.0</td>
<td>12.1</td>
<td>12.2</td>
<td>12.3</td>
<td>12.0</td>
<td>12.1</td>
<td>12.2</td>
<td>12.3</td>
<td>12.0</td>
<td>12.1</td>
<td>12.2</td>
<td>12.3</td>
<td>12.0</td>
<td>12.1</td>
<td>12.2</td>
<td>12.3</td>
</tr>
<tr>
<td>15</td>
<td>15.0</td>
<td>15.1</td>
<td>15.2</td>
<td>15.3</td>
<td>15.0</td>
<td>15.1</td>
<td>15.2</td>
<td>15.3</td>
<td>15.0</td>
<td>15.1</td>
<td>15.2</td>
<td>15.3</td>
<td>15.0</td>
<td>15.1</td>
<td>15.2</td>
<td>15.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B,D</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
</tr>
<tr>
<td>2</td>
<td>2.0</td>
<td>2.1</td>
<td>2.2</td>
<td>2.3</td>
</tr>
<tr>
<td>3</td>
<td>3.0</td>
<td>3.1</td>
<td>3.2</td>
<td>3.3</td>
</tr>
<tr>
<td>4</td>
<td>4.0</td>
<td>4.1</td>
<td>4.2</td>
<td>4.3</td>
</tr>
<tr>
<td>5</td>
<td>5.0</td>
<td>5.1</td>
<td>5.2</td>
<td>5.3</td>
</tr>
</tbody>
</table>

**Legend:**
- **p,q** and **B,D** represent indices for the ScaLAPACK library.
- The values in the table correspond to the block cyclic distribution of data in parallel computing applications.
Industrial Application of ScalAPACK

- Large Scale industrial application employed by Syracuse Research Corporation (SRC) in defense simulations of radar cross sections for “flying objects”

- Serial code (used LINPACK) widely used by SRC’s customers, but to allow simulation of new “flying objects” with a lot of mesh details necessary, HPCC was needed.

- Cost performance, portability across platforms was driving force. Code was sufficiently large that software investment effort porting to a single proprietary system was risky.

- Scalability also an issue for even larger problems in future.
SRC ParMoM Package

- Parametric Patch Method of Moments Code for radar cross section modeling of full airborne system.
- Problem can be summarised as assembly and solution of large dense matrix equation
- Matrix contains impedance coefficients
- RHS is (multiple) excitation vectors from different incoming radar signals
- solution vector is electric currents over surface of aircraft.
Design of Parallel ParaMoM

- Main component of the code is matrix L.U factorisation and solve (this is $O(N^3)$, where $N$ is number of unknown or the points for this application.)

- although some proprietary systems have library for this (eg Thinking Machines' CMSSL, or Intel SSL) ScaLAPACK was only truly portable one.

- Matrix assembly is $O(N^2)$ and disassembly is $O(N)$ which are still significant for very large $N$.

- ScaLAPACK is conveniently implemented on the BLACS layer, which was an appropriate communications library for the matrix assembly code. The interoperability of these two layers allowed a truly portable application code.
Parallel ParaMoM

- Successful ports to Intel (ScaLAPACK BLACS on NX/2); CM5 (using CMMD); IBM SP2 using EUI-H; various workstation clusters (Sun, DEC, IBM,...) using PVM as underlying layer, including use of underlying ATM hardware.

- tunable blocking parameters in ScaLAPACK library were valuable in tuning different application problem (mesh sizes) to different architectures - in a portable way.
Selected Timing comparisons for $N = 4889$ (in seconds)

<table>
<thead>
<tr>
<th>Platform</th>
<th>$N_p$</th>
<th>Fill</th>
<th>LU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha(FDDI)</td>
<td>8</td>
<td>1420</td>
<td>1120</td>
</tr>
<tr>
<td>IBM RS(Ether)</td>
<td>8</td>
<td>1501</td>
<td>1805</td>
</tr>
<tr>
<td>iPSC/860</td>
<td>64</td>
<td>526</td>
<td>281</td>
</tr>
<tr>
<td>CM-5</td>
<td>32</td>
<td>3295</td>
<td>171</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Platform</th>
<th>$N_p$</th>
<th>Setup</th>
<th>RHS</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>+ Field</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alpha(FDDI)</td>
<td>8</td>
<td>12.3</td>
<td>18.0</td>
<td>2570.2</td>
</tr>
<tr>
<td>IBM RS(Ether)</td>
<td>8</td>
<td>51.4</td>
<td>28.2</td>
<td>3385.0</td>
</tr>
<tr>
<td>iPSC/860</td>
<td>64</td>
<td>45.4</td>
<td>53.0</td>
<td>904.9</td>
</tr>
<tr>
<td>CM-5</td>
<td>32</td>
<td>21.1</td>
<td>4.3</td>
<td>3491.3</td>
</tr>
</tbody>
</table>

Portability and interoperability is greatest benefit.
Timing curves for various implementations:

Fig. 4 Fill Time vs. Processors (Matrix Size=988)

Fig. 5 LU Time vs. Processors (Matrix Size=988)
Conclusions/Summary:

- use of existing (tested) software always favored by industry

- cluster technology is viable for CEM applications of modest size

- use of portable (library based) software HPCC software allows straightforward move from application development on cluster to production run on MPP.

- good HPCC software libraries can be constructed - with careful design and high quality software engineering.

- final thought - software libraries may form major component of the runtime libraries for high level parallel languages such as HPF.
Online Internet Resources:

- [http://www.npac.syr.edu/](http://www.npac.syr.edu/) The Northeast Parallel Architectures Center (NPAC) Main Server (containing documentation on CEM Application of ScaLAPACK)

- [http://www.netlib.org/nse/home.html](http://www.netlib.org/nse/home.html) The National HPCC Software Exchange (containing ScaLAPACK software and documentation)

- Ken Hawick (hawick@npac.syr.edu); [http://www.npac.syr.edu/users/hawick/homepage](http://www.npac.syr.edu/users/hawick/homepage)

- David Walker (walker@msr.epm.ornl.gov); [http://www.epm.ornl.gov/walker](http://www.epm.ornl.gov/walker)