Porting a sphere optimization program from LAPACK to ScaLAPACK

Paul Leopardi

Mathematical Sciences Institute, Australian National University.
For presentation at Computational Techniques and Applications Conference
Australian National University, Canberra, July 2008.
Joint work with Rob Womersley, University of New South Wales.

July 2008

AUSTRALIAN RESEARCH COUNCIL
Centre of Excellence for Mathematics and Statistics of Complex Systems

Outline of talk

- The problem: maximizing a Gram determinant
- The approach: optimization using L-BFGS-B
- Converting the serial optimization code to ScaLAPACK
- Performance of the resulting parallel code

Maximizing a Gram determinant

Maximize the Gram determinant $\det \mathbf{G}$ (Sloan, Womersley, 2004).

- Gram matrix $\mathbf{G}$ of degree $\mathbf{n}$ is a function of a set of points $\{\mathbf{x}, \ldots, \mathbf{x_m}\}$ on the unit sphere $\mathbf{S^2}$, where $\mathbf{m} = (\mathbf{n}+1)^2$.

$$\mathbf{G_{i,j}} := \frac{\mathbf{n}+1}{4\pi}\mathbf{p}(\mathbf{x_i} \cdot \mathbf{x_j}),$$

  where $\mathbf{p} := \mathbf{P_n^{(1,0)}}$ is a Jacobi polynomial of degree $\mathbf{n}$.
- $\mathbf{G}$ is symmetric non-negative definite.
- If $\mathbf{G}$ is non-singular then $\{\mathbf{x_1}, \ldots, \mathbf{x_m}\}$ uniquely interpolates all spherical polynomials to degree $\mathbf{n}$.

Optimization using L-BFGS-B

Sphere optimization program SPHOPT uses L-BFGS-B to obtain a local maximum of $\det \mathbf{G}$.

- L-BFGS-B (Zhu, Byrd, Lu, Nocedal, 1994)
  - Based on Limited Memory BFGS (Nocedal, 1980; Liu, Nocedal, 1989),
  - Based on BFGS (Broyden-Fletcher-Goldfarb-Shanno, 1970) quasi-Newton optimization method.
- L-BFGS-B needs the function value and gradient at each step.

SPHOPT function value and gradient

- Function value $\mathbf{f = \log\det G}$ is obtained from the Cholesky decomposition $\mathbf{G = LL^T}$ via

$$\mathbf{f = 2\sum_{i=1}^{m} \log L_{i,i}.}$$

- Gradient $\mathbf{\nabla f}$ where $\mathbf{(\nabla f)_{k,i} := \frac{\partial f}{\partial X_{k,i}}}$ is computed by

$$\mathbf{\nabla f = 2X\,(DG \bullet G^{-1}),}$$

  where $\mathbf{X_{k,i} := (x_i)_k}$, $\mathbf{k = 1, 2, 3}$, $\mathbf{(DG)_{i,j} := \frac{n+1}{4\pi} p'(x_i \cdot x_j)}$ and $\bullet$ is the Hadamard product.
- SPHOPT uses LAPACK for Cholesky decomposition (DPOTRF), inverse (DPOTRI) and multiply (DSYMM).

ScaLAPACK

- Distributed memory parallel linear algebra (Choi, Dongarra, Pozo, Walker, 1992; Blackford et al. 1997).
- Distributed memory versions of LAPACK linear algebra routines, eg. dense solves, matrix inversion, eigensystems.
- Uses Block Cyclic data layout.
- Parallel Basic Linear Algebra Subroutines (PBLAS) includes matrix-vector and matrix-matrix products.
- Basic Linear Algebra Communications Subsystem (BLACS).
- Often implemented using Message Passing Interface (MPI) (Dongarra, Hempel, Hey, Walker, 1993).

PSPHOPT code structure using ScaLAPACK

- All processes run from the beginning of the one program.
- BLACS calls enable broadcast communication and synchronization between processes.
- To control loops and branches the PSPHOPT program:
  1. Sends all relevant data to a control process,
  2. Makes the decision in the control process,
  3. Broadcasts the decision.
- L-BFGS-B runs in the control process.
  1. Control process broadcasts the current point set $\mathbf{X}$,
  2. PSPHOPT uses ScaLAPACK to obtain $\mathbf{f}$ and $\nabla\mathbf{f}$,
  3. Control process calls L-BFGS-B with $\mathbf{f}$ and $\nabla\mathbf{f}$,
  4. L-BFGS-B calculates a new $\mathbf{X}$, or stops.

PSPHOPT code structure using ScaLAPACK

- ScaLAPACK calls need synchronization between processes.
- Structure of ScaLAPACK use is:
  1. Distribute operands,
  2. Synchronize,
  3. Operate,
  4. Distribute results.
- Gram matrix $\mathbf{G}$ is a function of the current point set $\mathbf{X}$.
  - Only $\mathbf{X}$ needs to be distributed per step.
  - Each process creates its own local parts of $\mathbf{G}$ and $\mathbf{DG}$.
- PSPHOPT uses ScaLAPACK for Cholesky decomposition (PDPOTRF), inverse (PDPOTRI) and multiply (PDSYMM).

Compressed block cyclic storage

- ScaLAPACK uses Block Cyclic data distribution to store arrays.
- ScaLAPACK routines on symmetric matrices touch only one triangle.

- PSPHOPT uses a square processor array. This simplifies storage and addressing of symmetric matrices **G** and **DG**.
- PSPHOPT uses the unused triangle of **G** to store most cycles of **DG**. The diagonal cycles of **DG** are stored in a separate array.

Compressed block cyclic storage of **G** and **DG**

	0	0	1	1	0	0	1	1
0	$G_{1,1}$	$G_{1,2}$	$G_{1,3}$	$G_{1,4}$	$G_{1,5}$	$G_{1,6}$	$G_{1,7}$	$G_{1,8}$
0	$G_{2,1}$	$G_{2,2}$	$G_{2,3}$	$G_{2,4}$	$G_{2,5}$	$G_{2,6}$	$G_{2,7}$	$G_{2,8}$
1	$G_{3,1}$	$G_{3,2}$	$G_{3,3}$	$G_{3,4}$	$G_{3,5}$	$G_{3,6}$	$G_{3,7}$	$G_{3,8}$
1	$G_{4,1}$	$G_{4,2}$	$G_{4,3}$	$G_{4,4}$	$G_{4,5}$	$G_{4,6}$	$G_{4,7}$	$G_{4,8}$
0	$DG_{1,5}$	$DG_{1,6}$	$DG_{1,7}$	$DG_{1,8}$	$G_{5,5}$	$G_{5,6}$	$G_{5,7}$	$G_{5,8}$
0	$DG_{2,5}$	$DG_{2,6}$	$DG_{2,7}$	$DG_{2,8}$	$G_{6,5}$	$G_{6,6}$	$G_{6,7}$	$G_{6,8}$
1	$DG_{3,5}$	$DG_{3,6}$	$DG_{3,7}$	$DG_{3,8}$	$G_{7,5}$	$G_{7,6}$	$G_{7,7}$	$G_{7,8}$
1	$DG_{4,5}$	$DG_{4,6}$	$DG_{4,7}$	$DG_{4,8}$	$G_{8,5}$	$G_{8,6}$	$G_{8,7}$	$G_{8,8}$

APAC SC nodes and interconnect

Australian Partnership for Advanced Computing (APAC) National Facility SC cluster(2001 to 2005):

- Compaq AlphaServer SC45 with 127 nodes each containing:
  - 4×1 GHz ev68 (Alpha 21264C) cpus
  - L1 cache (on chip): 64 kbytes (I) + 64 Kbytes (D)
  - L2 cache (off chip): 8 Mbytes per cpu
  - between 4 and 16GB of RAM

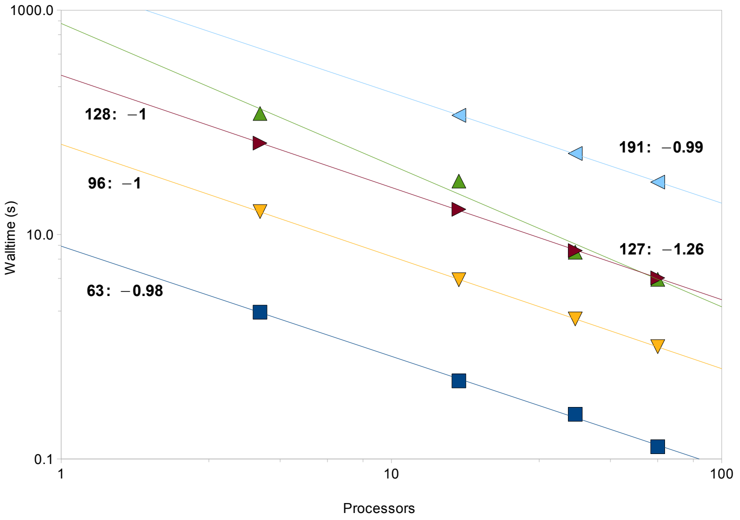
- Quadrics Elan3 interconnect:
  - MPI latency of $< 5\ \mu s$
  - MPI bandwidth of 250 Mbyte/s bidirectional

APAC AC nodes and interconnect

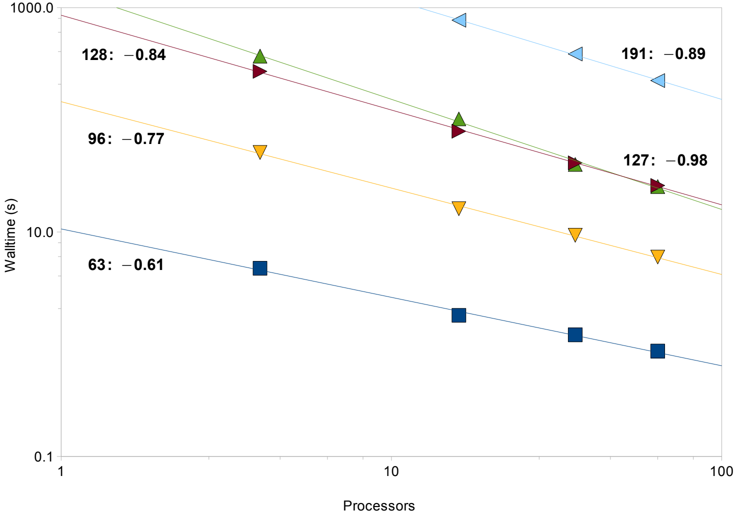
APAC National Facility AC cluster (2005 to present):

- SGI Altix 3700 Bx2 cluster with 30 nodes each containing:
  - 64 $\times$ 1.6 GHz Itanium2 cpus with:
    - L1 cache: 16 kbytes (D) + 16 kbytes (I). Cache line 64bytes
    - L2 cache: 256 kbytes. Cache line 128 bytes
    - L3 cache: 6 Mbytes. Cache line 128 bytes
  - between 128 GB and 384 GB of RAM
- SGI NUMAlink4 interconnect within and between nodes:
  - MPI latency of $< 2\ \mu s$
  - Bandwidth of 3.2 Gbytes/s bidirectional

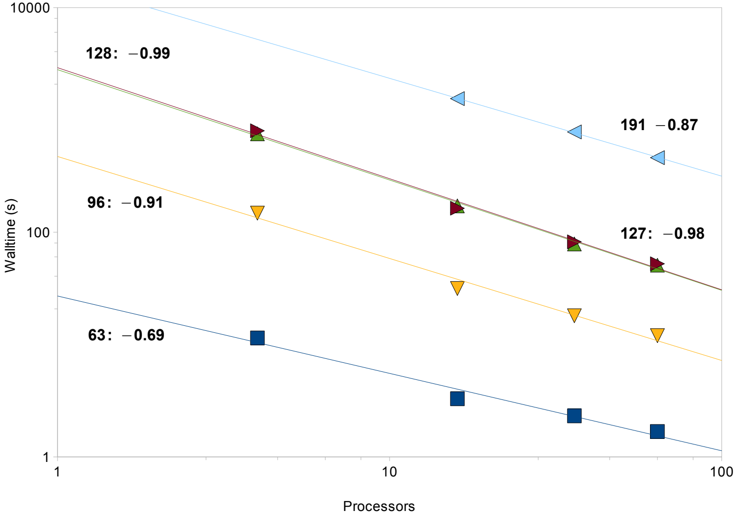
APAC SC: Gram matrix time



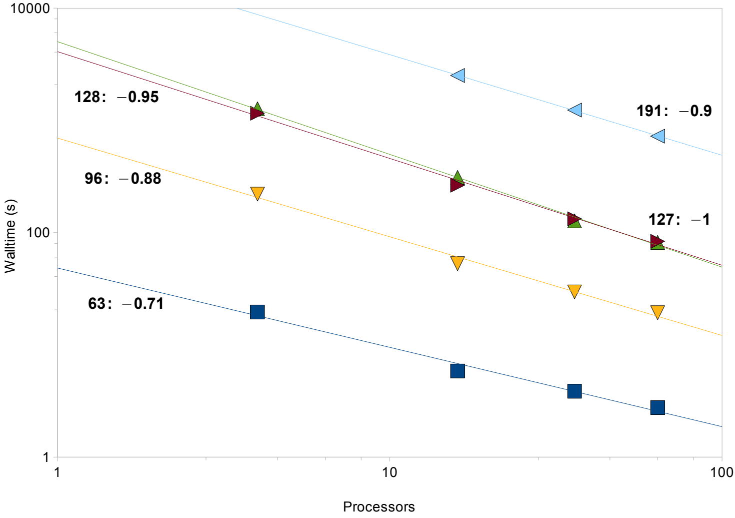
APAC SC: Cholesky factor (PDPOTRF) time



APAC SC: Cholesky inverse (PDPOTRI) time



APAC SC: Total $\mathbf{f}$ and $\nabla\mathbf{f}$ time



APAC AC: Total L-BFGS-B, $\mathbf{f}$ and $\nabla\mathbf{f}$ time