MATLAB on Multi-core Clusters

Yes, there is a parallel MATLAB
Is There Parallel MATLAB?

- Cleve's Corner by Cleve Moler
  - Title: *Why there isn't a parallel MATLAB*
    - Memory model
      - Distribution of data takes longer than computation
    - Granularity
      - Not much MATLAB's internal functionality can be made in parallel
  - Business situation
    - No customers with parallel computers
  - Year of publication: 1995
  - Machines of the time:
    - Ardent Titan
    - Intel iPSC (128 nodes)
- 2008: parallel MATLAB is a must
  - Clusters are everywhere
  - I cannot buy a single-core processors
Parallel MATLAB? Really?

- Programming environment for matrices
  - Programming language with \ as an operator

- Shell environment
  - System command shell
    - ! ls
  - GUI shell
    - fig = figure()
  - Java shell
    - str = java.lang.String('Hello world!')

- Set of toolboxes (80+)
  - SymBio, ...

- Parallel RAD IDE?
  - pafor, spmd, parallel job management, multi-cores, clusters
Overview

- Multithreading
- parfor keyword
- spmd keyword
- Parallel jobs
- Distributed arrays
Multithreading

- No changes to the code
- Control of parallelism
  - GUI menu
  - oldThreadCount = maxNumCompThreads(new)
- Functional scope
  - BLAS
  - LAPACK
  - Built-in operators
  - Built-in functions

- Test
  - for n = 1:1000
    - max(svd(randn(n)))
  - end
- Efficiency (lack of it)
  - 1 core 100%
  - 2 cores 52%
  - 3 cores 35%
  - 4 cores 26%

- Questions:
  - Why multithreading didn't work?
  - No multithreading in MATLAB?
  - Multithreading no good?
Overview

- Multithreading
- parfor keyword
- spmd keyword
- Parallel jobs
- Distributed arrays
### for → parfor

- **for** \( n = 1:1000 \)
  - \( \max(\text{svd}(\text{randn}(n))) \)
- **end**

- **Efficiency** (complete lack of it)
  - 1 core
    - 1 100%
  - 2 cores
    - 2 50%
  - 3 cores
    - 3 33%
  - 4 cores
    - 4 25%

- **parfor** \( n = 1:1000 \)
  - \( \max(\text{svd}(\text{randn}(n))) \)
- **end**

- **Efficiency**
  - 1 core
    - 1 100%
  - 2 cores
    - 2 89%
  - 3 cores
    - 3 83%
  - 4 cores
    - 4 79%
Example parfor Loop

• for j = 1:N
  – total1 = total1 + j
  – total2 = max(total2, foo(j))
  – total3 = bar(total3, j)
• end

• parfor j = 1:N
  – % operator as a reduction
  – total1 = total1 + j
  – total2 = max(total2, foo(j))
  – total3 = bar(total3, j)
• end
Parallel for Loops with `parfor`

- Minimal changes to code for `parfor`
- Control of parallelism
  - `matlabpool()`
- Random order of iterations
  - Helps load balancing
- Built-in and custom reductions
  - Heavy code analysis
- Requirements
  - Iteration independence
  - Code transparency
- Different than OpenMP
  - No need for shared memory
  - No need for special syntax/pragmas
Overview

- Multithreading
- parfor keyword
- spmd keyword
- Parallel jobs
- Distributed arrays
parfor $\rightarrow$ spmd

- parfor n = 1:1000
  - max(svd(randn(n)))
- end

- spmd
  - for n = 1000:100000
    - cdstr = codistributor
    - A = randn(n, cdstr)
    - max(svd(A))
  - end
- end
spmd: Remoteness and Persistence

- Automatic variable classification
  - In = 1
  - InOut = 3
  - spmd
    - Out = 2
    - InOut = In + Out
  - end
- Automatic data transfer
- Persistence
  - spmd
    - x = 1
  - end
  - z = 2
  - spmd
    - y = x + z
  - end
spmd and Composites

- Composites
  - \( A = \text{rand}(1000) \)
  - \( \text{class}(A) \% 'double' \)
  - spmd
    - \( \text{svd}(A) \)
  - end
  - \( \text{class}(A) \% 'Composite' \)

- Remote reference
  - No data transfer unless explicitly dereferenced

- Cell-like interface
  - clientA = A{1}
spmd Varieties

- spmd
  - x = numlabs;
- end
- spmd(3)
  - x = numlabs;
- end
- spmd(3, 6)
  - x = numlabs;
- end
- spmd(0, Inf) same as spmd
  - x = numlabs;
- end
Controlling Resources with Matlabpool

- Matlabpool syntax – quick reference
  - `matlabpool open`
  - `matlabpool close`
  - `matlabpool size`
  - `matlab open local 3`
  - `matlab open MyCluster 127`

- Controls toolboxes that already use parfor
Overview

- Multithreading
- parfor keyword
- spmd keyword
- Parallel jobs
- Distributed arrays
Parallel Jobs: Syntax

- `mpirun -machine MyCluster`
- `-np 127`
- `ModelSimulation.exe inputDataFile outputDataFile`
- `qsub ModelSimulation.pbs`
- `while [true]`
  - `if [qstat] break`
- `end while`
- `sched = findResource('Conf', 'MyCluster')`
- `job = createParallelJob(sched)`
- `set(job, MinimumNumberOfWorkers, 127)`
- `task = createTask(job, @ModelSimulation, 1 inputData)`
- `submit(job)`
- `waitForState(job, 'finished')`
- `oargs = getAllOutputArguments(job)`
Parallel Jobs: Overview

- Schedulers and batch systems
  - local
    - For a laptop, desktop, single multi-core node
  - Job Manager
    - Allows callbacks from cluster
  - PBS Pro, Torque
  - Platform LSF
  - Sun Grid Engine
  - Windows CCE: CCS1, CCS2
  - mpiexec
    - Shell command: mpirun, mpiexec, ...
  - <generic>
    - Condor, ...

- Debugging
  - Deadlock detection

- Profiling
  - mpiprofile() function
    - Switch on or off
  - Not based on PMPI layer

- Swappable MPI implementation
  - Must be MPICH2 binary compatible
    - HP
    - Intel
    - Microsoft
    - MVAPICH2
    - Myricom
  - MPI 3.0 request: ABI for MPI
MPI vs. MATLAB

- MPI_Comm_rank
- MPI_Comm_size
- MPI_Send
- MPI_Recv
- MPI_Sendrecv
- MPI_Barrier
- MPI_Broadcast
- MPI_Probe
- MPI_Reduce(..., MPI_SUM)
- MPI_Reduce(..., MyFunction)
- mpirun -machinefile MyCluster
- xterm -e MySimulation.exe

- labindex
- numlabs
- labSend
- labReceive
- labSendReceive
- labBarrier
- labBroadcast
- labProbe
- gplus
- gop
- pmode start MyCluster
pmod

- Parallel shell
- Think: “inside spmd”
  - spmd
    - <parallel shell>
  - end

- MATLAB look-and-feel
- Ideal for
  - Prototyping
  - Debugging
Overview

- Multithreading
- parfor keyword
- spmd keyword
- Parallel jobs
- Distributed arrays
Distributed Arrays: Quick Look

- Sequential code
  - \( N = 1000 \)
  - \( A = \text{rand}(N) \)
  - \( b = \text{rand}(N, 1) \)
  - \( t = \text{tic} \)
  - \( x = b \div A \)
  - \( t = \text{toc}(t) \)
  - \( \text{fprintf}(1, 'Gflop/s=%g', 2/3 * N^3 / t) \)

- Parallel code
  - \( N = 100000 \)
  - \( A = \text{rand}(N, \text{codistributor}) \)
  - \( b = \text{rand}(N, 1, \text{codistributor}) \)
  - \( t = \text{tic} \)
  - \( x = b \div A \)
  - \( t = \text{toc}(t) \)
  - if (labindex == 1)
    - \( \text{fprintf}(1, 'Gflop/s=%g', 2/3 * N^3 / t) \)
  - end
Distributed Arrays: Functionality

- Many overloaded methods
  - 150+ functions
  - Operators
  - Linear algebra
  - Indexing
  - Data analysis

- Distribution schemes
  - Variant 1D
    - codistributor('1d', dim, partition)
  - 2D block cyclic
    - codistributor('2d', [labRows labCols], blkSize)