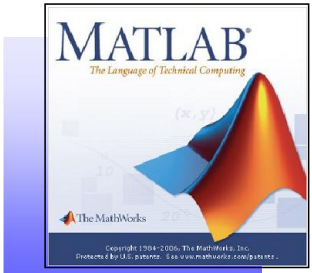
High Performance and GPU Computing in MATLAB

Jan Houška

HUMUSOFT s.r.o.

houska@humusoft.cz

http://www.humusoft.cz



About HUMUSOFT

Company: ***Humusoft s.r.o.***

Founded: ***1990***

Number of employees: ***18***

Location: ***Praha 8, Pobřežní 20***

The MathWorks

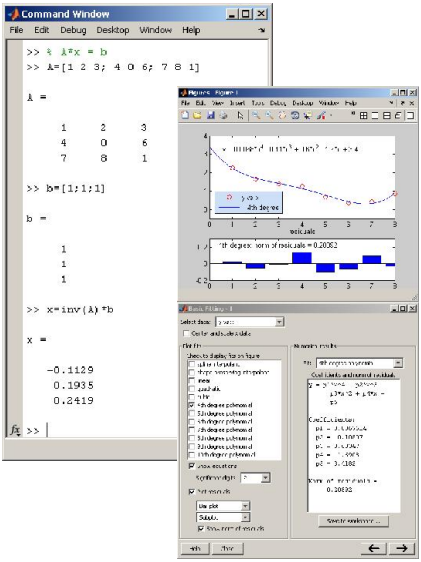
COMSOL

dSPACE

- **MATLAB, Simulink**
- **Comsol Multiphysics**
- **dSPACE development systems**
- **HeavyHorse multiprocessor workstations**
- **Training, consulting, development**

Computations in MATLAB

- **The basic data type is a matrix**
  - algorithms optimized to work with vectors and matrices
  - vector and matrix operations
  - systems of linear equations
  - polynomials and data fitting
  - statistic functions
  - trigonometric functions
  - …
  - discrete Fourier transform
  - ordinary differential equations
- **More than 1000 functions from different areas available**
  - elementary math functions



How do I accelerate my computations?

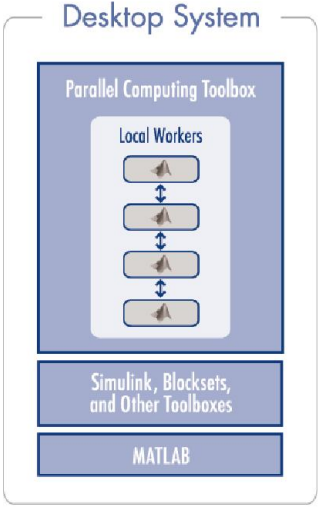
- **Q: I don't see any significant speedup on my high-end multi-core workstation**
  - **Why are most of my CPU cores idle?**
  - **How do I utilize my GPU?**
- **A: The algorithm must be parallelized to utilize multiple cores**
  - **computations must be at least partially independent**
  - **there must be no or little data dependency between concurrently running tasks**
- **Some algorithms are inherently parallel**
  - **little or no additional work required**
    - many MATLAB functions that work on individual vector and matrix elements
  - **good candidates for GPU acceleration**
- **Some algorithms are parallelizable, but not parallelized**
  - **various amount of additional work needed**
  - **candidates for explicitly parallel jobs**
- **Some algorithms are not parallelizable at all**
  - **the previous step of algorithm needs to be fully finished before the next step can start**
    - many ODE solvers
  - **there is no way to benefit from multiple cores**

Multiprocessing in MATLAB

- **Built-in multithreading**
  - built into core MATLAB
  - for specific vector and matrix operations
  - automatically enabled, no extra work necessary
- **Parallel programming**
  - division to tasks controlled by MATLAB programmer
    - various levels of control, from semi-automatic to fully controlled
  - code is run on multiple CPU cores
  - suitable for generic parallel computing
    - moderate number of tasks of arbitrary complexity
- **GPU computing**
  - code is run on a separate GPU device with dedicated memory
    - data needs to be transferred to GPU and back
  - suitable for massively parallel computing
    - very high number of simple tasks

Parallel Computing Toolbox

- **Design and implementation of parallel algorithms**
- **Structure**
  - **client**
    - MATLAB commands for job and task creation
  - **local scheduler**
    - distributes tasks to workers, gathers job results
  - **worker**
    - computational unit for a task
- **Up to 8 workers on a local machine**
  - **easily scalable to cluster of arbitrary size**
    - using MATLAB Distributed Computing Server
- **GPU hardware interface**
  - **NVidia CUDA**
    - compute capability 1.3 and higher required



Graphics Processing Unit (GPU)

- **Originally for graphics acceleration, now also used for scientific calculations**
- **Massively parallel array of integer and floating point processors**
  - **Typically hundreds of processors per card**
  - **GPU cores complement CPU cores**
- **Dedicated high-speed memory**

Parallel Computing Toolbox – GPU functions

- **Interface for GPU computing**
  - **math functions implemented on GPU**
  - **running MATLAB code on GPU**
  - **interface for running CUDA code from MATLAB**
    - argument passing to/from MATLAB
- **New since MATLAB Release 2010b**
  - **initial version of the interface**
  - **extensions expected in future versions**
    - new release of MATLAB is available twice per year
- **Typical applications**
  - **acceleration by parallel processing on the GPU**
  - **CUDA code development and debugging**

Parallel Computing Toolbox – GPU functions

- **Object `gpuDevice`**
  - identifies and selects the device
  - one GPU device per one MATLAB instance can be used
  - up to 8 devices per machine using parallel workers
  - `gpuDeviceCount`

```
>> gpuDevice
Properties:
                      Name: 'GeForce GTX 460'
                     Index: 1
         ComputeCapability: '2.1'
            SupportsDouble: 1
             DriverVersion: 3.1000
        MaxThreadsPerBlock: 1024
          MaxShmemPerBlock: 49152
        MaxThreadBlockSize: [1024 1024 64]
               MaxGridSize: [65535 65535]
                 SIMDWidth: 32
               TotalMemory: 1.0417e+009
                FreeMemory: 580116480
       MultiprocessorCount: 7
      GPUOverlapsTransfers: 1
    KernelExecutionTimeout: 1
          DeviceSupported: 1
           DeviceSelected: 1
```

Parallel Computing Toolbox – GPU functions

- **Object `gpuArray`**
  - **„handle“ to GPU data in MATLAB**
  - **math operations defined directly on the object**
    - indexing, multiplication, `abs`, `sin`, `floor`, `max`, `fft`, `lu`, `gamma`, `erfc`, ...
  - **`gather` transfers data back to MATLAB**
  - **supports real and complex numbers, different data types**
    - data types can differ in speed

```
magic(4);
xg = gpuArray(x);
rg = xg*xg;
r = gather(rg);
```

- **Data can be created directly on the GPU**
  - **no need for transfer from MATLAB**
  - **`parallel.gpu.GPUArray.zeros(1000)`**

Parallel Computing Toolbox – GPU functions

- **Running MATLAB code on GPU – function `arrayfun`**
  - applies the function to each array element
  - function can have multiple input and output arguments
  - automatically translates MATLAB code to GPU
  - supports a subset of MATLAB language

```
function [z, w] = sqrtsincos(x, y)
z = sqrt(sin(x)*cos(y));
w = sqrt(sin(y)*cos(x));

a = rand(4096); b = rand(4096);
ag = gpuArray(a); bg = gpuArray(b);
rg = arrayfun(@sqrtsincos, ag, bg);
r = gather(rg);
```

Parallel Computing Toolbox – GPU functions

- **Running CUDA code from MATLAB – object `parallel.gpu.CUDAKernel`**
  - directly runs CUDA PTX kernel
  - automatic conversion of input arguments
  - output arguments are `gpuArray` objects

```
__global__ void sqrtSinCos(double* v1, const double* v2)
{
  int idx = blockIdx.x*blockDim.x + threadIdx.x;
  v1[idx] = sqrt(sin(v1[idx])*cos(v2[idx]));
}
```

```
sqk = parallel.gpu.CUDAKernel('sqrtSinCos.ptx', 'sqrtSinCos.cu');
a = rand(1024, 1);
b = rand(1024, 1);
sqk.ThreadBlockSize = 1024;
rk = feval(sqk, a, b);
r = gather(rk);
```

Humusoft HeavyHorse

- AMD Opteron processors
  - two or four processors
  - 8 to 48 cores
  - CPU frequency 2.2 to 3.1 GHz
- 8 to 128 GB RAM
- Graphics accelerator NVIDIA Tesla C2050
  - supports GPU computing
  - available as an option
- Choice of operating systems
  - Microsoft Windows 64-bit: XP, Vista, 7, Server
  - Linux 64-bit: OpenSUSE, Ubuntu, ...
- Application software optionally pre-installed
  - MATLAB
    - Parallel Computing Toolbox
  - MATLAB Distributed Computing Server
  - COMSOL Multiphysics

Additional Information

- **Web sites**
  - **www.humusoft.cz**
    - homepage of Humusoft s.r.o.
  - **www.mathworks.com**
    - homepage of MathWorks
- **MATLAB Central**
  - **discussions, blogs, file exchange, questions & answers, ...**
    - www.mathworks.com/matlabcentral/
- **International Conference "Technical Computing Prague 2011"**
    - www.humusoft.cz/akce/matlab11
- **Discussion groups**
  - **Czech and Slovak MATLAB Users Group (CSMUG)**
    - www.humusoft.cz/produkty/matlab/csmug
  - **Usenet News**
    - comp.soft-sys.matlab