Hydraulic groundwater modeling

- Week 8

- The hardware side of numerical groundwater modeling

Why worrying about performance?

- Simulation of the real world requires enormous computational resources
- Serial processors are obsolete after some time, because there is a physical limit to processor speed (speed of light)
- Upgrading to parallel hardware by adding more processors
- Instead of specially developed single processors use many off-the-shelf (cheap) processors in parallel (e.g. through linux clusters)

The hardware

- For modeling we are interested in:
  - Processor
  - RAM
  - Hard Disk drive
  - Power Supply
  - Cooling
- Additionally for high-performance:
  - Video card (GPU)
  - Network (Ethernet, Infiniband,...)

Power Supply & Cooling

- Power supply
  - Converts alternating current (AC) to direct current (DC)
  - Performance needs to be sufficient to power all ingredients
  - (Often first part to get broken)

- Cooling
  - Usually based on fans (air)
  - large PCs have several separate fans (power supply, CPU, GPU)

Central processor unit (CPU)

- „computers brain“
- Consists of „cores“: Separate cores can process information simultaneously
- More cores = more tasks can be completed simultaneously
- Hyperthreading: Simulating more cores than there are physically
  - execution of code simultaneously on a single core due to idle times
  - In high-demanding, well-parallelized simulations this is a disadvantageous
- A CPU speed is expressed in cycles (GHz)

Persistent data storage

- Measures:
  - Capacity (GB/TB)
  - speed (revolutions per minute rpm)
- HDD:
  - magnetic storage with moving parts (rotating disk)
- SSD
  - integrated circuit to store data.
  - no moving parts and mechanically more robust
  - much faster in reading / writing operations than HDD but also more expansive

Random access memory (RAM)

- Temporary storage for data accessed by the CPU
- Parameters, variables initiated during a simulation will be stored here
- If required memory exceeds available RAM
  - Data will be stored on HDD/SDD, which is comparably slow
  - Software crash possible
- Memory management is a crucial part of simulation

Mainboard

- Main circuit containing most of the connectors
- „heart of the PC“
- Accommodates the Central Processor Unit (CPU), RAM slots, fan, …
- Limiting factor of RAM that can be added
- Limiting factor for network and GPU connections
- Determines communication speed between components

Integers and floating point arithmetic

- Integer:
  - Integral values = whole numbers (like 1 2 3 0 -1 -2 -3)

- Float:
  - Floating point values = values that have **potential** decimal places (like 3.142 or 3.000 or -3.0)

Bits and bytes

- A number is usually stored in 32 bits
- A bit can have the value 0 or 1
- Integer:
  - 1 bit for the sign (+/-) and 31 bits for the value
  - Resulting value range: -2^{31} – $(2^{31}-1)$
- 1 Byte = 8 bit
  - 32 bits = 4 byte

Machine accuracy

- Float
  - 1 bit for the sign, 8 bits for the exponent, 23 bits for the mantissa
  - Maximum value is around $3.4 \cdot 10^{38}$

$$\pm M \cdot 2^{E}$$

  - A 32-bit float has only a precision of 6 to 7 decimal digits
    - You can represent $4.51523 \cdot 10^{35}$ or $3.12489 \cdot 10^{-27}$ but all further digits are not specified by the float

Assessing memory demand

- Assume a regular grid
  - Store two parameters (transmissivity and storativity)
  - Solve for one variable (hydraulic head)
- Grid size 200 x 100 grid points = 20 000 entries
- For all stored values: 20 000 x 3 = 60 000 entries
- All filled with 32-bit floats: 60 000 x 32bit = 19 200 000 bits
- Divide by 8 to obtain bytes = 2 400 000 bytes
- Use unit prefix = 2.4 MB

Assessing memory demand – ctd.

- Remember:
  - Regular grids require less memory than unstructured grid
  - The example of 3 values to be stored is an absolute minimum
  - Solving the equation will require more memory

- Nowadays, the 32-bit is called „single precision“
- Often 62-bit (16 decimal digits) are used – called „double precision“

Computational costs

- In computer science: many definitions, rules of estimation,…
  - Associated with ‚computational complexity‘
- Here:
  - Execution time per time step during simulation
  - memory demands
- Costs usually increase with problem/input size (in bits)
  - More grid points = more calculations (& maybe smaller time step)

Assessing code performance

- Rules of thumb:
  - Integer operations are easier than floating point arithmetics
  - Addition and subtraction are easier than multiplication and division
  - Reduce number of calculations necessary
    - Store often used combinations of variables in memory
- Reduce output
  - Accessing graphics or hard drive slows down simulation
- Reduce unnecessary computational load
  - Close non-used programs and windows

Towards high-performance computing (HPC)

- Also called „super-computing“
- In general associated with HPC centers around the world
- Performance measured in „floating point operations per second“ (FLOPS)
- Basically: A network of extremely powerful PCs with a really fast connection between them
- Used for various scientific applications, also in Geosciences!
- HPC techniques can be helpful at small scale networks as well
- First step: What are your computational requirements ?

Parallel programming

- The fundamental laws of physics are parallel in nature
  - They apply at all times at each point in space.
- The state of a physical quantity in the near future depends on
  - its present state
  - its immediate past
  - its nearest neighborhood
- The most important task in parallel programming:
  - Which part of the problem can be efficiently parallelized?

Seriel vs. Parallel programming

- Serial style:
  - One processor which is executing a series of instructions
  - There is a logical sequential flow through the program
  - At any time there is only one operation being carried out by the processor
- Parallel computing:
  - Producing the same result using multiple processors
  - The problem is divided between a number of processors
  - Functional decomposition (multi-tasking)
  - Data decomposition (data-parallel)
  - Keep all the processors busy (load-balancing)

Parallelization techniques

- Shared memory:
  - Separate CPU cores access the same RAM → one large / powerful PC
- Distributed memory:
  - Not all involved CPU cores access the same RAM. Usually a network of „regular“ PCs.
  - If several CPU cores work at the same problem, they need to communicate

Shared memory / Multiprocessing

- As all cores have all information, parallelization is rather simple
- If implemented, cores share automatically the load (not the domain!)
- A well parallelized program has all cores under full load
- Number of involved CPU cores also called „threads“
- Do not allow more „threads“ than you have physical cores (do not use Hyperthreading)

- Common technique used by most software nowadays

Distributed memory

- Pieces of information have to be passed from one thread to another
  - Message passing (via network)
- Fast threads might need to wait for slower threads to communicate
  - blocking communication
- Assigned tasks for each core can not be easily altered during runtime
- Timing and amount of communication is important
- Large problems will require distributed memory
- Much more scalable than shared memory computation

Graphical Processor Units (GPU)

- Originally designed to rapidly manipulate data for visual output
  - video games and realistic optical presentation
- Calculating quickly large amount of data is attractive for simulations
- Each GPU core is slower than a CPU core but there are hundreds of it
- Graphics memory similar size than RAM
- GPUs fail if unbalanced load over cores (branch prediction)
- A GPU always needs a CPU as controlling and communication unit

Lessons learned

- Hardware management is crucial!
  - Processor
  - RAM
  - Data storage
- Estimating hardware demands
- Modern software will apply HPC-techniques
  - Parallelization techniques
  - Shared/distributed memory