Design large scale Research(Science) and Engineering Software

Volume I: fundamental concept

Volume II: software engineering

Volume III: code analysis

1. **Introduction**
    1. 1. The challenge of big data
        1. 1. 1. scenarios:
        1. 1. 2. intelligent CAD and CAE
    1. 2. Modern PC hardware
        1. 2. 1. Increasing computation power by Moore's law
        1. 2. 2. Quantum computer
    1. 3. Interaction with physical world
        1. 3. 1. design prototype, CAD
        1. 3. 2. optimisation of existing
        1. 3. 3. digital twin
    1. 4. Are you using software correctly?
        1. 4. 1. integer arithmatic overflow
        1. 4. 2. floating point overflow
        1. 4. 3. process the data properly
    1. 5. Audience of this book

2. **Infrastructure for high-performance computation**
    2. 1. Parallel computation within CPU:
        2. 1. 1. **find out your CPU**
        2. 1. 2. computation power
        2. 1. 3. Inter-computation unit connection
    2. 2. 64-bit computing
        2. 2. 1. have you installed a 64bit Operation system
        2. 2. 2. data type in programming language
            2. 2. 2. 1. integer range
            2. 2. 2. 2. Unsigned integer
            2. 2. 2. 3. deal with algorithm overflow
        2. 2. 3. float or decimal
            2. 2. 3. 1. sequence of float operation is important for precision
        2. 2. 4. float point arithmetic exception
        2. 2. 5. Migrate to 64bit computation
    2. 3. GPU
        2. 3. 1. GPU: super-computer in one die
        2. 3. 2. **find out your GPU**
        2. 3. 3. CPU-GPU communication
        2. 3. 4. supported hardware acceleration
    2. 4. Memory technology CPU-thirsty
        2. 4. 1. lagging memory speed
        2. 4. 2. DDR technology
        2. 4. 3. Multiple-channel technology
        2. 4. 4. GDDR

2. 4. 5. CPU cache
2. 4. 6. Memory access and allocation is expensive
2. 5. Super computer archicture
2. 5. 1. Physical arrangement of super computer
2. 5. 2. inter-node networking
2. 5. 3. Institutional HPC
2. 6. IO Speed
2. 6. 1. PCIe IO bus
2. 6. 2. IO speed limited by system call
2. 7. The power of parallel computation
2. 7. 1. OS support for parallel computation
2. 7. 1. 1. process management
2. 7. 1. 2. IPC
2. 7. 1. 3. threading
2. 7. 2. asynchronous programming
2. 7. 3. Multi-threading and thread pool
2. 7. 3. 1. OpenMP
2. 7. 4. MPI for coupled data
2. 7. 5. GPU computation
2. 7. 6. Parallel IO
2. 8. scheduling supercomputer
2. 8. 1. software compiled for a specific HPC
2. 8. 2. resource management
2. 8. 3. job submission
3. **Infrastructure for big data**
3. 1. big data in research and engineering
3. 2. big object in memory
3. 2. 1. VM and program memory layout
3. 2. 2. create a large object in memory
3. 2. 3. data structure for big data
3. 3. Storage for super computer
3. 3. 1. External non-volatile storage
3. 3. 2. Storage tech and speed
3. 3. 2. 1. Storage connection for PC
3. 3. 3. Storage server for super computer
3. 3. 3. 1. shared-disc FS
3. 3. 3. 2. distributed FS
3. 4. local file systems
3. 4. 1. file system types
3. 4. 2. local storage area network (SAN)
3. 5. distributed file system
3. 5. 1. storage network array
3. 5. 2. NFS
3. 5. 3. cloud FS
3. 6. Cross platform data format
3. 6. 1. OS-independent file path
3. 6. 2. textual file and encoding
3. 6. 3. binary file and CPU endianness
3. 7. file format for large dataset
3. 7. 1. HDF5
3. 7. 2. netCDF v4
3. 7. 3. XDMF Reader
3. 7. 4. XML partitioned VTK file
3. 8. Parallel FileSystem and MPI-IO

- 3. 9. Database for big data application
  - 3. 9. 1. Conventional RDB
  - 3. 9. 2. In-memory DB
  - 3. 9. 3. SQLite the single-file database

4. **Problem scale-up and partitioning**
- 4. 1. estimate problem scale
- 4. 2. strategy for paralell computation
  - 4. 2. 1. segment big data
  - 4. 2. 2. partitioning a large scale problem
  - 4. 2. 3. geomtry decomposation
- 4. 3. divide and conquer
- 4. 4. Communication and collaboration
  - 4. 4. 1. introduction to IPC
  - 4. 4. 2. Message Passing Interface (MPI)
  - 4. 4. 3. DDS
  - 4. 4. 4. Kafka, RabbitMQ, AMQP, Mqtt, JMS

5. **Parallel algorithm and concurrnecy**
- 5. 1. race condition and synchronization
  - 5. 1. 1. atomics
  - 5. 1. 2. C++11 threading
- 5. 2. concurrent data structure
  - 5. 2. 1. lock and syn

6. **The jungle of programming language**
- 6. 1. Introduction
  - 6. 1. 1. Timeline of programming languages
  - 6. 1. 2. How to select and learn your language
- 6. 2. Compiling lang for HPC
  - 6. 2. 1. Fortran:
  - 6. 2. 2. C/C++
  - 6. 2. 3. evoluaiton of C++
    - 6. 2. 3. 1. evolution of C++
  - 6. 2. 4. Compiling process of C/C++
  - 6. 2. 5. The design of LLVM and GCC
  - 6. 2. 6. Java, JVM and JIT
  - 6. 2. 7. C# and dotnet framework
  - 6. 2. 8. Other compiling languages
- 6. 3. interpreting lang
  - 6. 3. 1. Introduction
  - 6. 3. 2. Python: the language for data science
  - 6. 3. 3. Tcl/TK: science
- 6. 4. languages for computation
  - 6. 4. 1. Matlab
  - 6. 4. 2. Other general
- 6. 5. web programing
  - 6. 5. 1. web scripting
  - 6. 5. 2. data spec spec HDML5
  - 6. 5. 3. webbrowser as a VM
- 6. 6. mixed language programming
  - 6. 6. 1. Mixing C, Fortran and C++
  - 6. 6. 2. language wrapping
  - 6. 6. 3. dotnet CLR and Java VM
  - 6. 6. 4. language independent interface

6. 7. language binding for Python
6. 7. 1. write python module in C or C++
6. 7. 2. Boost.python and PyBind11
6. 7. 2. 1. Cython: write python module in C++
6. 7. 3. cppyy: JIT and binding generation
6. 7. 4. SWIG for python
6. 7. 5. Fortran to Python
6. 7. 6. Qt and GTK's own wrapping
7. **Architecture for Cross-platform software**
7. 1. Cross-platform software design
7. 1. 1. Cross-platform
7. 1. 2. Linux Fragmentation and LSB
7. 1. 3. Operation systems and compilers
7. 1. 3. 1. Is there any library to do that?
7. 1. 4. cross-platform building system
7. 1. 5. cloud computation
7. 1. 6. challenging of testing
7. 2. Sustainable Component selection
7. 2. 1. lifecycle plan
7. 2. 2. Key components
7. 2. 3. Tools selectoin
7. 3. API design
7. 3. 1. consistant naming convention
7. 3. 2. function design
7. 3. 3. class API design
7. 3. 4. API document
7. 4. ABI and API compatibility
7. 4. 1. binary compatible is crucial for enterprise platforms
7. 4. 2. Compiler linkage: static or shared?
7. 4. 3. Find the correct shared library
7. 4. 4. libraries version control
7. 4. 5. ABI and forward compatibility
7. 4. 6. C/C++runtime
7. 4. 7. C++ pImpl Idiom for stable ABI
7. 4. 8. API stability
7. 5. Modular design
7. 5. 1. module (java package) level encapsulation
7. 5. 2. binary plugin design
7. 5. 2. 1. portable binary plugin system
7. 6. Extensible architecture
7. 6. 1. source code level extension
7. 6. 2. bindary level extension
7. 6. 3. protocol based extensible framewrok
7. 7. Accessible User interface
7. 7. 1. TUI or GUI
7. 7. 2. web UI and restful API
7. 7. 3. Human-brain VR AI
8. **Refactor legacy project**
8. 1. Reason, tools for refactoring
8. 1. 1. why needed
8. 1. 2. process
8. 1. 3. tools for refactoring
8. 2. Porting to another platform
8. 3. Redesig and rewrite

9. Large Software Project Management
9. 1. proposal and funding
9. 1. 1. funding source for initative
9. 1. 2. long term community-driving
9. 2. Software license
9. 2. 1. Open source software license
9. 2. 2. Documentation license
9. 2. 3. The Creative Common Licenses
9. 3. Community dvelopment
9. 3. 1. One dominant
9. 3. 2. elected committee
10. Productivity and Quality control
10. 1. software engineering models
10. 2. Source management system
10. 2. 1. Git for version control
10. 2. 2. Efficient team collaboration
10. 3. Software testing
10. 3. 1. unit test and coverage
10. 3. 2. regression
10. 3. 3. integration test
10. 3. 4. physical validation
10. 4. Software productivity tools
10. 4. 1. IDE tools
10. 4. 2. other tools
10. 5. Continuous integration (CI)
10. 5. 1. improve compiling performance
10. 5. 2. automated and parallel testing
10. 5. 3. Container for different platforms
10. 6. Code quality and code style
10. 6. 1. code style or smell
10. 6. 2. const exception thread-safety contract
10. 6. 3. code analysis tools
10. 7. Documentation
10. 7. 1. generation from source code
10. 7. 2. structure
10. 7. 3. book, wiki and forum
11. Debugging, Profiling and Optimization Tools
11. 1. debugging
11. 1. 1. debugger
11. 1. 2. tools to discover potential bugs
11. 2. Profiling/benchmarking
11. 2. 1. computation time and memory usage
11. 2. 2. profiling tools
11. 2. 2. 1. `perf` is the modern tool
11. 2. 2. 2. igprof
11. 2. 2. 3. sprof
11. 2. 3. benchmarking tools and methods
11. 3. optimization
11. 3. 1. compiler optimisation
11. 3. 2. code analysis by trace
12. Release and packaging
12. 1. predictable and frequent release

12. 2. packaging on Linux
12. 2. 1. Linux package formats and official repository
12. 2. 2. AppImage, snap and flatpak
12. 2. 3. Docker image and cloud computation
12. 3. windows portable app or installer
12. 4. Post-release: Bug tracking
13. **Good practice to design large C++ Software**
13. 1. have you really master C++
13. 1. 1. function signature overloading
13. 1. 2. keyword `static` `using`
13. 2. use modern C++11
13. 2. 1. git rig of raw pointers by smart pointers
13. 2. 1. 1. new smart pointers
13. 2. 1. 2. avoid using reference by `shared_ptr`
13. 2. 1. 3. Return only value type of smart pointers
13. 2. 1. 4. pass shared smart pointers as function parameter
13. 2. 1. 5. be careful to common errors using smart ponters
13. 2. 1. 6. make_shared(T) or shared_ptr(new T())
13. 2. 1. 7. thread safety of smart pointers
13. 2. 1. 8. STL iterator is pointer typedef
13. 2. 1. 9. std::any as a better std::shared_ptr
13. 2. 2. `std::function` and functional programming
13. 2. 3. constexpr
13. 2. 4. Type traits and template enhancement
13. 3. C++20 and beyond
13. 3. 1. module
13. 3. 2. parallel TBB
13. 4. understandable: design patterns
13. 5. extensible: modulerisation
13. 5. 1. example by KDE5 tier hierachy
13. 6. reliable: testing
13. 6. 1. unit test, encapsuation
13. 6. 2. feature/functional test (integration test)
13. 6. 3. coverage
13. 6. 4. physical testing /market validation
14. **Efficient Python programming**
14. 1. join effort by Python
14. 2. search instead of reinventing the wheel
14. 3. fast prototyping
14. 4. Version and runtime
14. 4. 1. Is your python import the correct module?
14. 4. 2. python in the cloud
14. 5. documentation
14. 5. 1. versatile doxygen
14. 5. 2. Sphinx and ReST
14. 6. build your own swiss knife kit
15. **Workflow automation by shell script**
15. 1. the power of batch processing
15. 1. 1. why shell script in 21 centry?
15. 1. 2. other scripting language
15. 1. 2. 1. Python
15. 2. shell scripting

15. 2. 1. learn bash script in one day
15. 2. 2. pitfalls of shell script
15. 2. 3. minimal requirement

16. **Computational Mathamatics**

16. 1. linear algebra
16. 1. 1. LINPACK and LAPACK
16. 1. 2. PETSc numps
16. 2. Numerical method ODE and PDE
16. 3. Computational geometry
16. 3. 1. Computational geometry kernels
16. 3. 2. Open source libraries
16. 3. 3. OpenCASCADE
16. 4. Topology and Graph theory
16. 4. 1. networkX for python
16. 4. 2. boost::graph for C++
16. 5. statistics and probability
16. 6. stochastic methods
16. 6. 1. monta-carlo methods
16. 7. misc
16. 7. 1. Symbolic math
16. 7. 2. crypto

17. **Scientific software**

17. 1. method, spatial and temporal scale
17. 2. Mesoscale simulation
17. 2. 1. LBM
17. 2. 2. Monta-Carlo
17. 3. Molecular dynamics
17. 3. 1. Lammps
17. 4. Quantum mechanics
17. 4. 1. Plasma physics: Bout++, CFD

18. **Open source Computer-aided engineering (CAE)**

18. 1. CAD
18. 1. 1. Open source CAD
18. 1. 2. data exchange STEP 242
18. 2. Partitioning
18. 2. 1. ParMETIS 1
18. 2. 2. SCOTCH and PT-SCOTCH 1
18. 2. 3. Hypre
18. 3. meshing or pre-processor
18. 3. 1. Meshing methods and file formats
18. 3. 2. Netgen and GMSH
18. 3. 3. SALOME and smesh
18. 4. FEA: Dolfin (FEniCS)
18. 5. CFD: OpenFOAM
18. 6. Visualization
18. 6. 1. The design of paraview
18. 6. 2. OSPRay
18. 7. Optimization
18. 8. misc
18. 8. 1. Dimension analysis, units

19. **Big data and AI**