VPS and Scalability on ARM

Arm HPC User's Group at ISC'19

Greg Samonds and Thorsten Queckboerner
ESI Group

VPS and Scalability on ARM

Agenda

- Overview and introduction
  - VPS – Major characteristics and typical applications
  - Tuning activities – A historical overview
  - HPC relevant code characteristics
- VPS explicit on ARM
  - Compiler version comparison – Runtime and compilation speed
  - Compiler options and runtime
  - Scalability VPS explicit on ARM

VPS and Scalability on ARM

Agenda

- Overview and introduction
  - **VPS – Major characteristics and typical applications**
  - Tuning activities – A historical overview
  - HPC relevant code characteristics
- VPS explicit on ARM
  - Compiler version comparison – Runtime and compilation speed
  - Compiler options and runtime
  - Scalability VPS explicit on ARM

VPS – Virtual Performance Solution

What our Customers are doing

- Characteristics of VPS (formerly also called PAM-CRASH)
  - Structural mechanics FE solver
  - Explicit solution scheme
  - Implicit solution schemes
  - General purpose Code with a strong focus on Automotive Virtual Prototyping

- Typical Customer applications
  - Crash and Safety simulation
  - Dummies and Human models
  - Airbag unfolding with gas dynamics (embedded FPM CFD solver)
  - Static load cases
  - Implicit dynamics, NVH, acoustics

VPS – Virtual Performance Solution

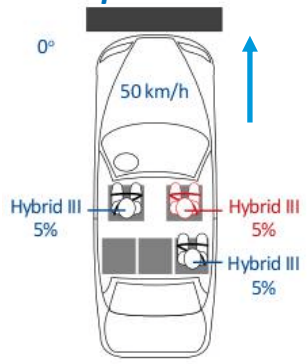
Virtual **P**rototyping - Crash Analysis

VPS – Virtual Performance Solution

Virtual Prototyping – Safety and Dummies

- Objective
  - Virtual assessment of injury risks and safety ratings
  - Realistic representation of deformation, injury, kinematics and sensor readings needed
- Dummy development and calibration
  - Set up of experimental program for calibration of numerical model
  - Validation of material, component and full body

Test setup example



Source: https://www.safetywissen.com

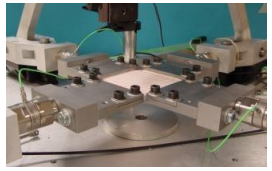
Injury risk example

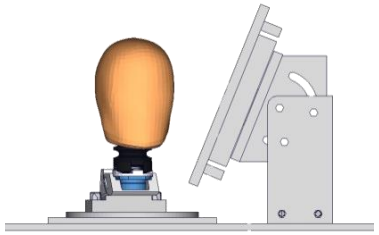
Source: http://www.mynrma.com.au/motoring-services/reviews/ancap/small-cars/mitsubishi-mirage-2013-ancap.htm

Safety rating example

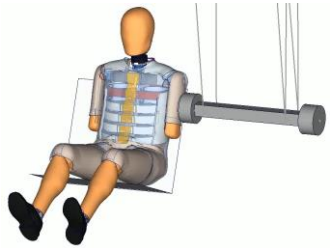
Source: http://www.euroncap.com/en

Material tests

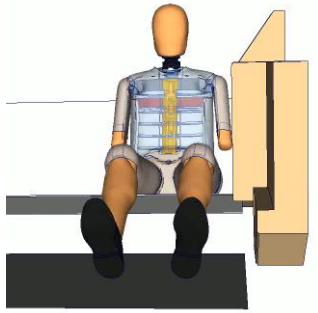


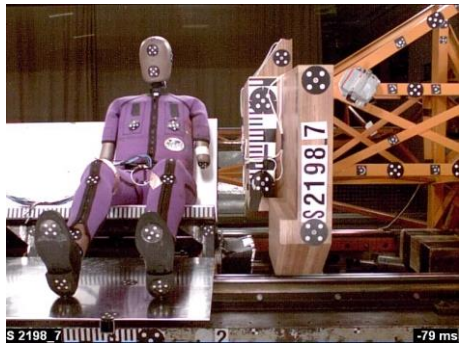


Component tests



Full body pendulum tests

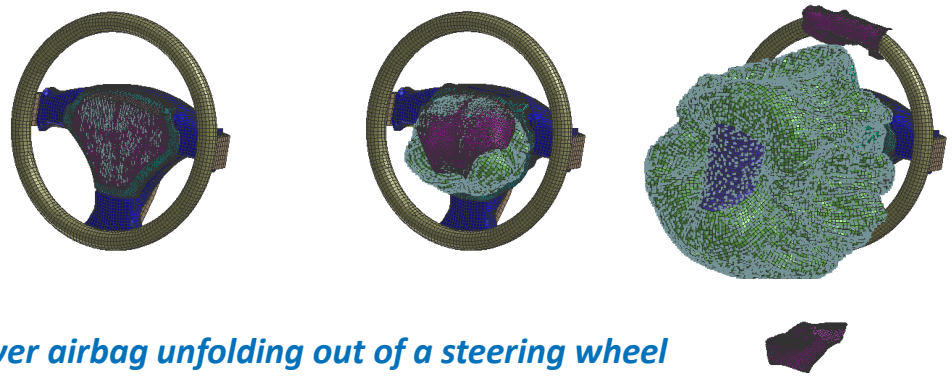




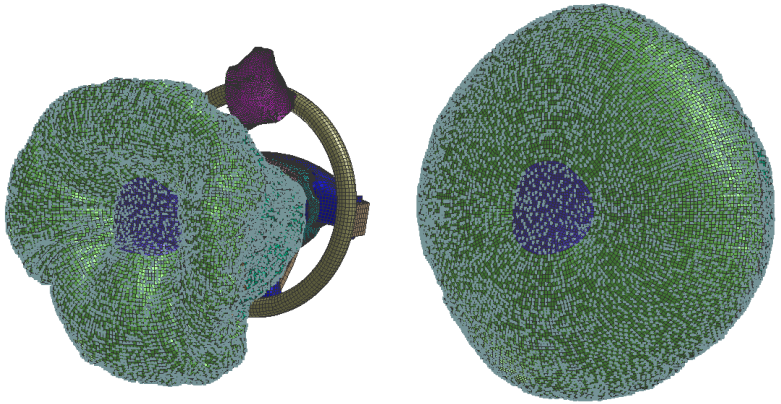
Full body barrier tests

VPS – Virtual Performance Solution

Virtual **P**rototyping - Airbag Modelling



Driver airbag unfolding out of a steering wheel



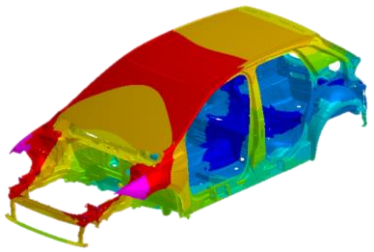
Driver airbag unfolding out of a steering wheel (cut view)
FPM velocity vectors

VPS – Virtual Performance Solution

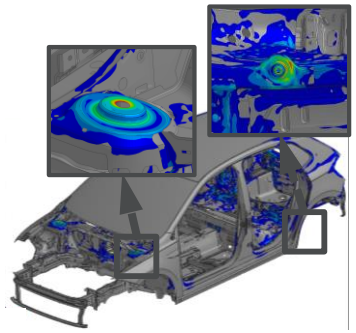
Virtual **P**rototyping - Statics & Dynamics & Acoustics

- Results

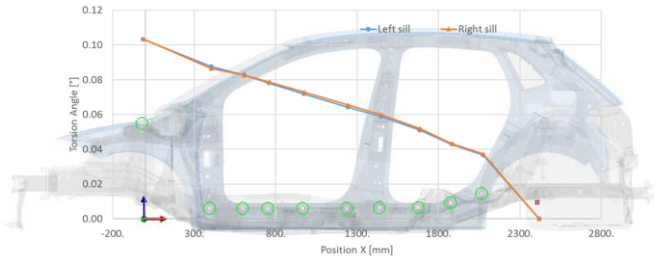
Static loading



Displacements

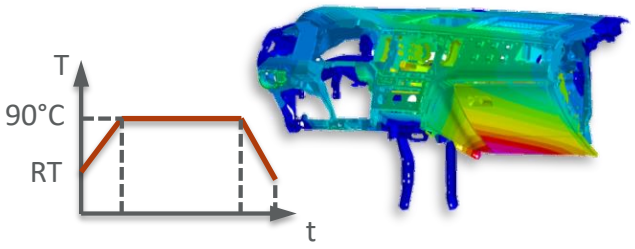


Stresses



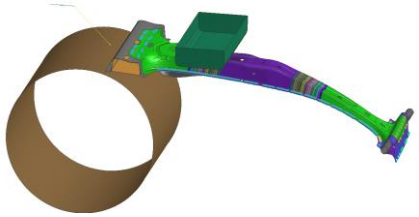
Torsion angle along vehicle length

Creep



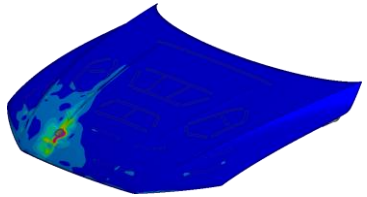
Remanent deformation after climate chamber

Bending Test



Deformation/Strength

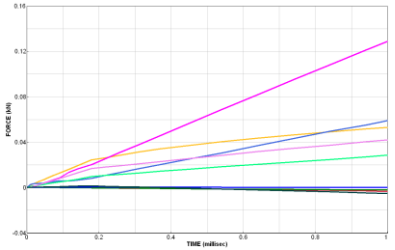
Indentation Test (Oil-Canning)



Buckling/ Remaining deformation

Misuse

Stresses/ Remaining deformation



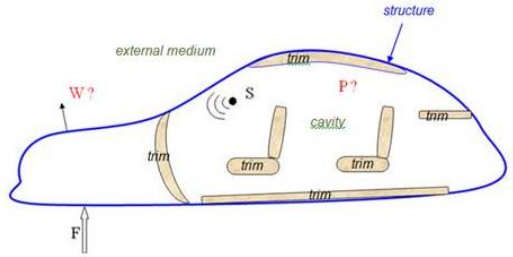
Forces of retainer clips

VPS – Virtual Performance Solution

Virtual Prototyping - Statics & Dynamics & Acoustics

DYNAMICS

- Objective
  - Improve structural dynamic strength
  - Increase comfort during operation
    - Noise
    - Vibrations

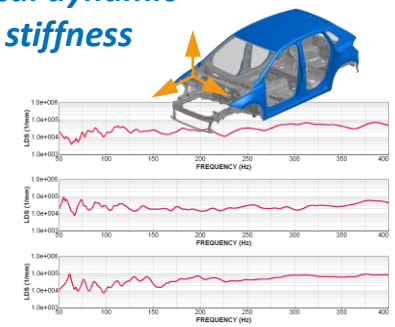


- Methodology
  - Linear dynamics
    - Structural vibrations (FE)
    - Acoustics in/outside of cavity (FE/BEM)
    - Coupled vibro-acoustics (FE-FE, FE-BEM)

Eigenfrequencies/-modes

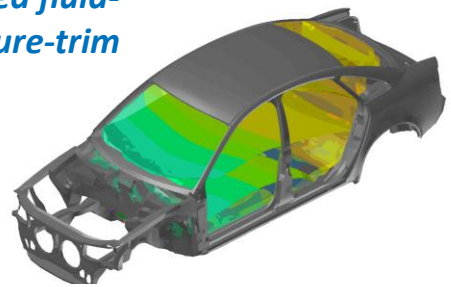
Resonance frequencies

Local dynamic stiffness

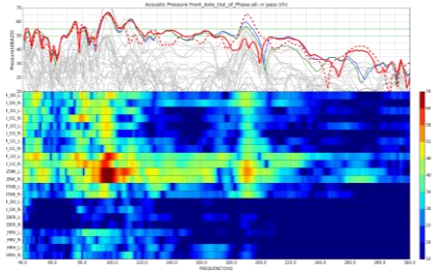


Frequency dependent stiffness

Coupled fluid-structure-trim



Interior/exterior sound pressure level

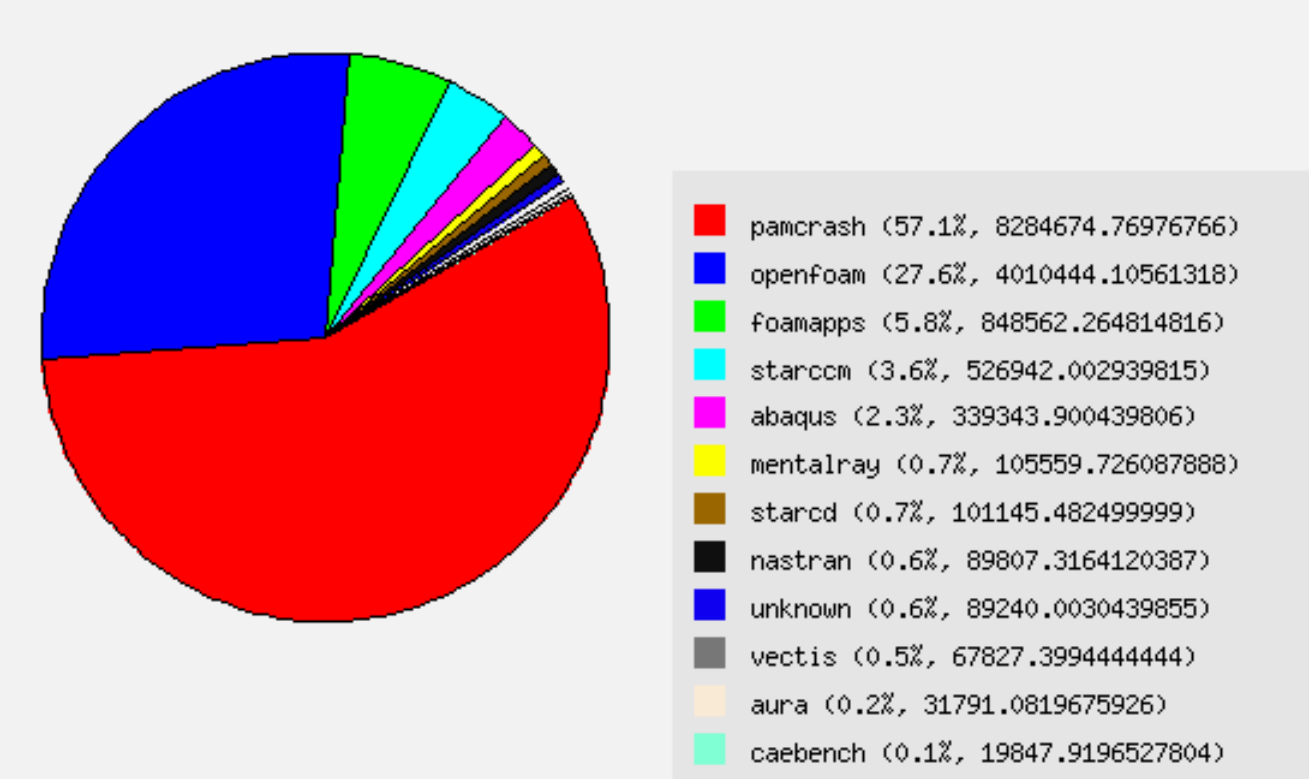


Transfer path analysis

VPS – Virtual Performance Solution

Usage at Customers/HPC-Centers/Partners

- Automotive OEM: HPC usage (12 months):
  - VPS-explicit 57 %
  - openfoam 28 %
  - foamapps 6 %
  - starccm 4 %
  - abaqus 2 %
  - mentalray < 1 %
  - starcd < 1 %
  - nastran < 1 %
  - vectris < 1 %
  - aura < 1 %
  - caebench < 1 %



VPS and Scalability on ARM

Agenda

- Overview and introduction
  - VPS – Major characteristics and typical applications
  - **Tuning activities – A historical overview**
  - HPC relevant code characteristics
- VPS explicit on ARM
  - Compiler version comparison – Runtime and compilation speed
  - Compiler options and runtime
  - Scalability VPS explicit on ARM

VPS – Virtual Performance Solution

Tuning Activities

- Constantly increasing requirements with regard to computing load
  - Model size increase
  - More load cases
  - New applications
  - …
- Therefore, code tuning is daily business
  - Since more then 30 years
  - On different areas
    - Sequential
    - Vector
    - SMP - OpenMP
    - DMP - MPI
  - With all major HPC vendors/players (alphabetic order)
    - CRAY, COMPAQ, Fujitsu, IBM, HP, SGI, SUN, NEC …
    - AMD, ARM, INTEL …

VPS – Virtual Performance Solution

Tuning Activities

- Sequential tuning
  - Support new chip features
    - Vector computer
    - AVX units (x86)
    - Code modifications + Compiler features
  - Memory access
    - Tuning on cache usage
    - Direct access and data alignment
  - Algorithms
    - Faster algorithms
    - But keep memory consumption in mind

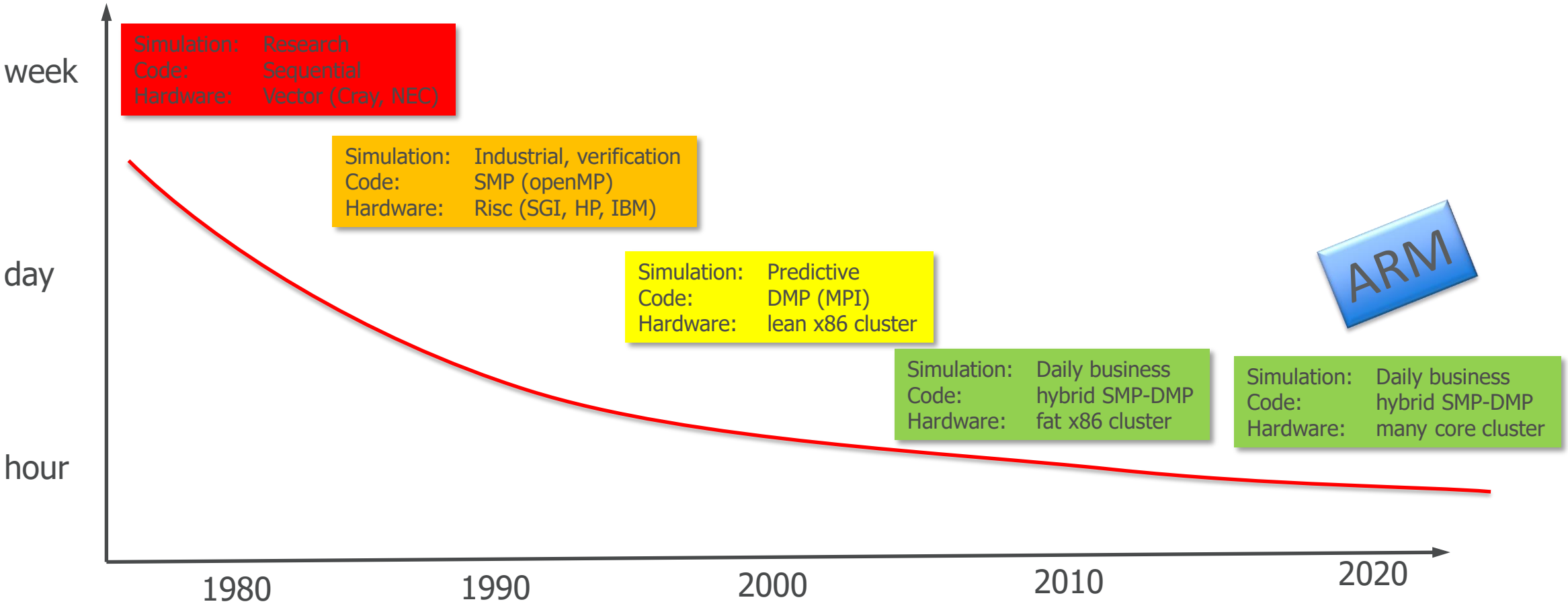
VPS – Virtual Performance Solution

Tuning Activities

- SMP tuning (OpenMP)
  - Load distribution
    - Thread data grain size
    - Thread load balance
  - Reduce/avoid locks
  - Thread and NUMA aware memory management
- DMP tuning (MPI)
  - Minimize number of communications: Interconnect latency
  - Minimize size of messages: Interconnect bandwidth
  - Minimize load unbalance
    - Static load unbalance (Domain decomposition)
    - Dynamic load unbalance (Open issue)

VPS – Virtual Performance Solution

Evolution: Simulation, Software and Hardware evolution (explicit)



VPS and Scalability on ARM

Agenda

- Overview and introduction
  - VPS – Major characteristics and typical applications
  - Tuning activities – A historical overview
  - **HPC relevant code characteristics**
- VPS explicit on ARM
  - Compiler version comparison – Runtime and compilation speed
  - Compiler options and runtime
  - Scalability VPS explicit on ARM

VPS – Virtual Performance Solution

Portability

- Language:
  - Std. Fortran (90 or higher)
  - Std. C
- Size (Solver core only):
  - 8000 files
  - 2.5 Million lines of code
- Supports all major OS
  - Unix, Linux, Windows
- Supports all major architecture
  - ia32, em64t, amd64, ia64, mips, pa-risc, ibm-power, ARM (work in progress)
- MPI Supports by dynamic load module
  - One executable – multiple MPI runtimes
- All major interconnect
  - Ethernet, Myrinet, Infiniband, Omnipath

VPS – Virtual Performance Solution

Performance characteristics

- Main code sections (for typical explicit crash/safety simulation):
  - Internal forces (FE) 50-65%
    - Cache friendly (scatter-gather to local arrays)
    - Many operations with few global data
    - Mainly direct addressing on static data
    - Scales very well for SMP and DMP
  - Explicit time integration 15-20%
    - Few and cheap operations on large global arrays
    - Direct addressing on static data
    - Most demanding option for memory bandwidth
    - Scalability in DMP and SMP limited by hardware
  - Contact 15-40%
    - Indirect addressing on unstructured dynamic data
    - Difficult for SMP, very difficult for DMP
    - Scales reasonable well for DMP and SMP

VPS – Virtual Performance Solution

Performance characteristics

- A Crash/Safe simulation model is a big challenge for HPC and parallelization:
  - Very feature rich and inhomogeneous input definition
    - All kind of elements
    - All kind of material models
    - Complicated contact
    - Particle methods (FPM, SPH)
    - Exotic safety options (slipring, airbag, MBSYS, …)

    -> **Flat profile – no real hotspots**

  - Big risk to hit
    - Load unbalance (dynamic or static)
    - Serial code (keep in mind Amdahl)

VPS and Scalability on ARM

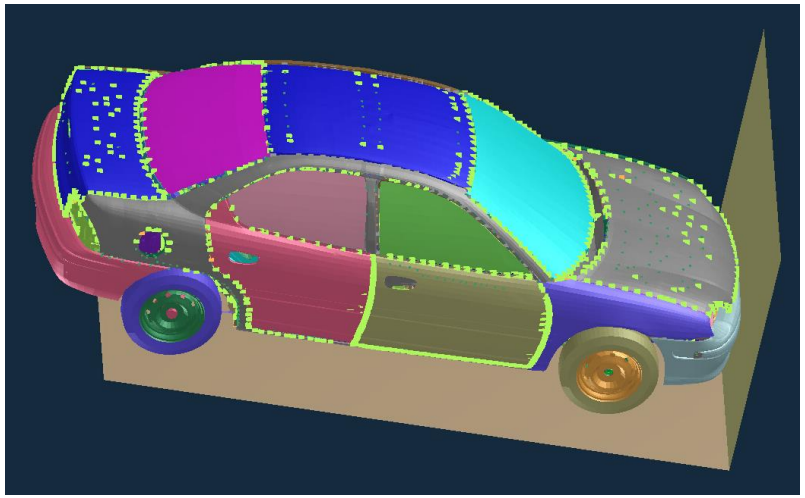
Agenda

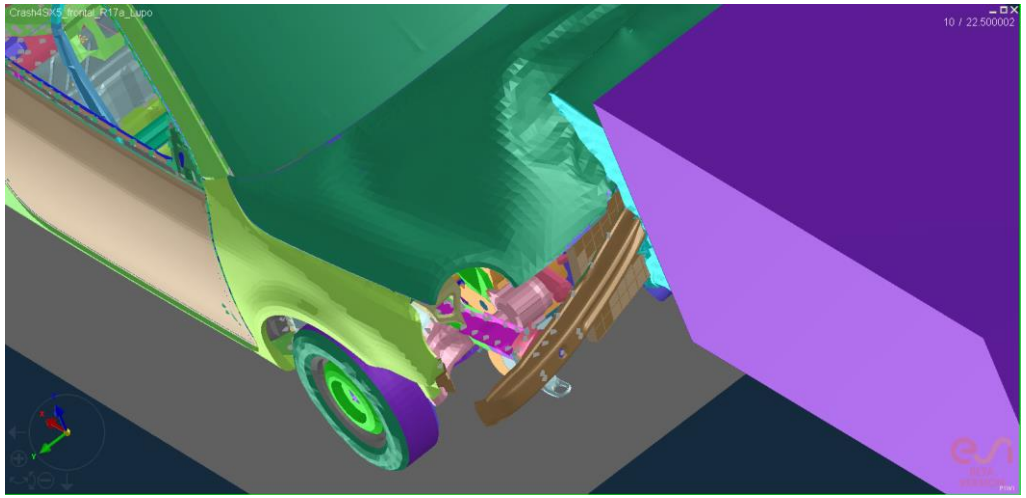
- Overview and introduction
  - VPS – Major characteristics and typical applications
  - Tuning activities – A historical overview
  - HPC relevant code characteristics
- **VPS explicit on ARM**
  - Compiler version comparison – Runtime and compilation speed
  - Compiler options and runtime
  - Scalability

VPS Arm Porting and Compilers Study

Testing the latest arm64 compilers, options, and scalability

- Compilers: ArmCC-19.0, ArmCC-19.1, ArmCC-19.2, GCC-8.3.0, GCC-9.1.0
- Options: Flags recommended by PRACE Best Practices Guide, flags recommended by hardware provider
- Scalability: Up to 96 cores
- Test cases: Neon and Lupo explicit crash simulation





Test Setup

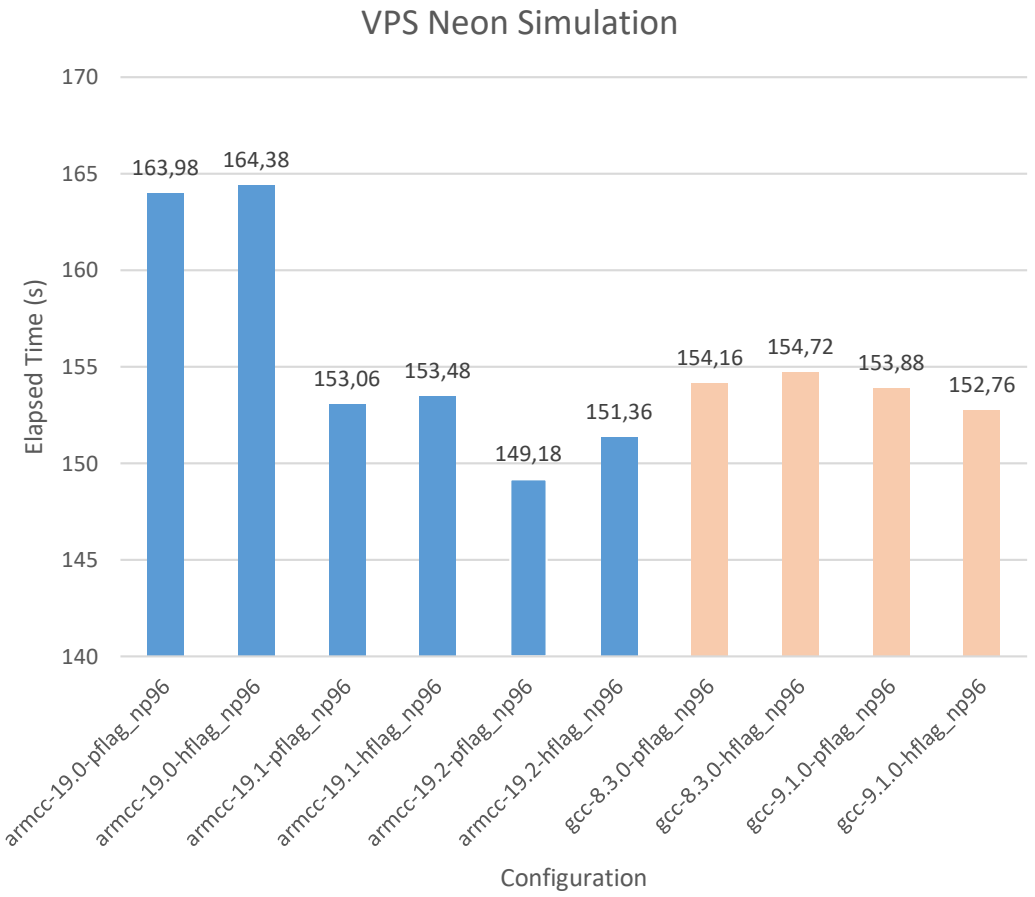
Machine configuration and compiler options

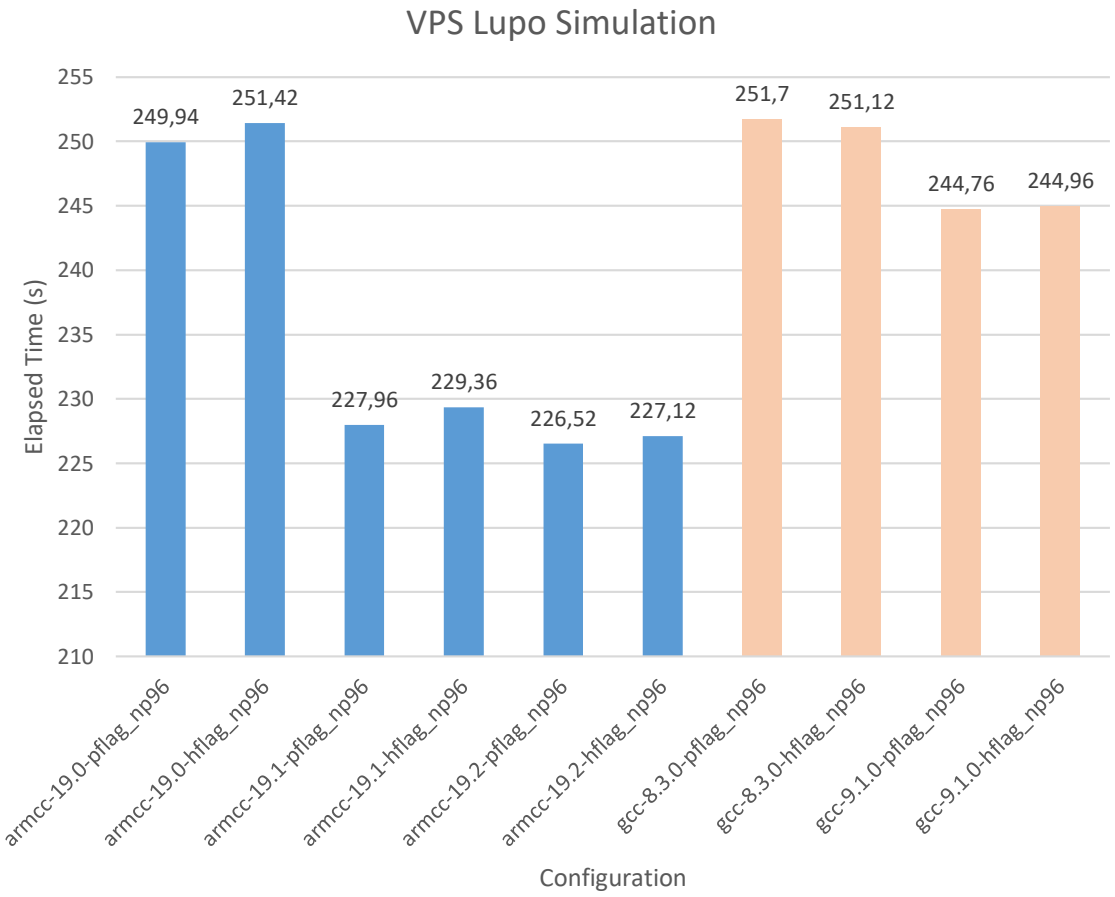
CPU	RAM	OS / Kernel	MPI Implementation	VPS Version
2 X Hi1620 48p (Kunpeng 920)	256GB DDR4 2666MHz	EL 7.6 / 4.14.0-115	HPC-X 2.3.0 (OpenMPI 4.1.0a1)	2018

Configuration	Options
armcc-19.0-pflag	"-march=armv8.2-a -mcpu=cortex-a72 -O3 -fcray-pointer -fopenmp -fomit-frame-pointer"
armcc-19.0-hflag	"-march=armv8.2-a+lse+crc+crypto+fp+simd -mcpu=cortex-a72 -O3 -ffp-contract=fast -funroll-loops -fcray-pointer -fopenmp -fomit-frame-pointer"
armcc-19.1-pflag	"-march=armv8.2-a -mcpu=tsv110 -O3 -fcray-pointer -fopenmp -fomit-frame-pointer"
armcc-19.1-hflag	"-march=armv8.2-a+lse+crc+crypto+fp+simd -mcpu=tsv110 -O3 -ffp-contract=fast -funroll-loops -fcray-pointer -fopenmp -fomit-frame-pointer"
armcc-19.2-pflag	"-march=armv8.2-a -mcpu=thunderx2t99 -O3 -fcray-pointer -fopenmp -fomit-frame-pointer"
armcc-19.2-hflag	"-march=armv8.2-a+lse+crc+crypto+fp+simd -mcpu=thunderx2t99 -O3 -ffp-contract=fast -funroll-loops -fcray-pointer -fopenmp -fomit-frame-pointer"
gcc-8.3.0-pflag	"-march=armv8.2-a+crc+simd -mtune=cortex-a72 -O3 -fcray-pointer -fopenmp -floop-optimize -falign-loops -falign-labels -falign-functions -falign-jumps -fomit-frame-pointer"
gcc-8.3.0-hflag	"-march=armv8.2-a+crc+simd -mtune=cortex-a72 -O3 -fcray-pointer -fopenmp -fbacktrace -ffpe-summary=none"
gcc-9.1.0-pflag	"-march=armv8.2-a -mtune=tsv110 -O3 -fcray-pointer -fopenmp -floop-optimize -falign-loops -falign-labels -falign-functions -falign-jumps -fomit-frame-pointer"
gcc-9.1.0-hflag	"-march=armv8.2-a+crc+simd -mtune=tsv110 -O3 -fcray-pointer -fopenmp -fbacktrace -ffpe-summary=none"

Test Results

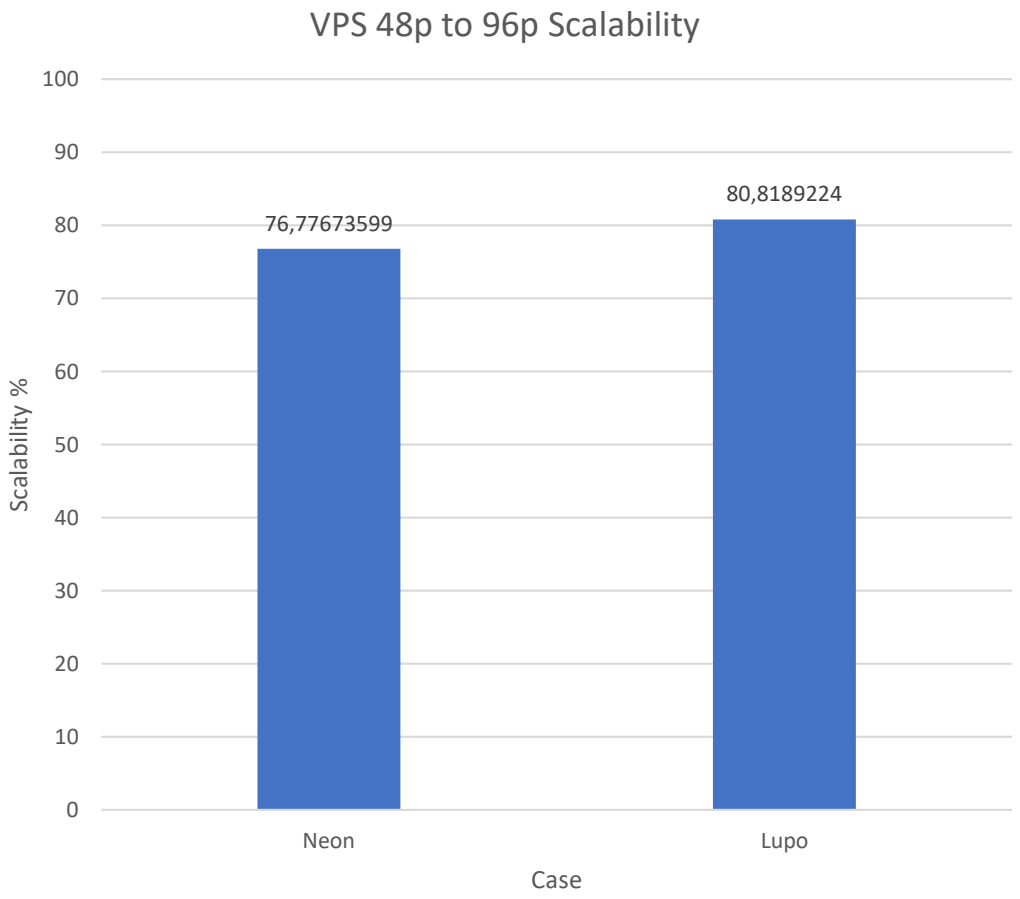
Compiled executable runtime comparison, 96p, 5 run average

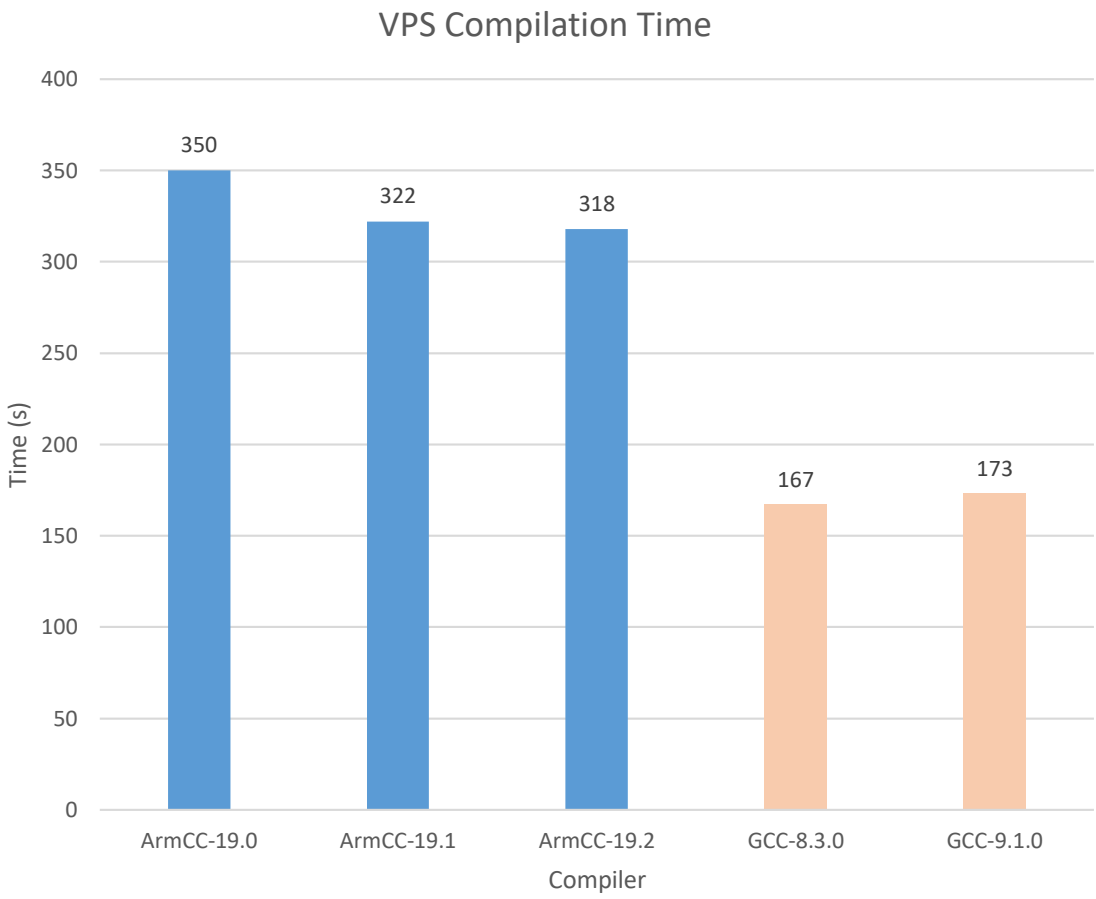




Test Results

Scalability and compilation speed





Summary

Observations from experiment

- Meaningful improvement in runtime with ArmCC-19.1/19.2 compared to 19.0
- ArmCC 19.2 executable was faster than GCC 9.1.0, with the margin depending on test case (Neon ~2%, Lupo ~8%)
- Significantly faster compilation speed with GCC compared to ArmCC
- Very small difference in runtime across tested compiler option sets, usually ~1% (within scatter)
- Scaling is limited by our machine's memory bandwidth. Previously observed ~85% (+10%) scaling up to 96 procs with Neon case on machine with faster RAM
- Promising overall performance, Huawei machine produced similar runtime to dual socket Xeon Gold 6150 machine

Thanks