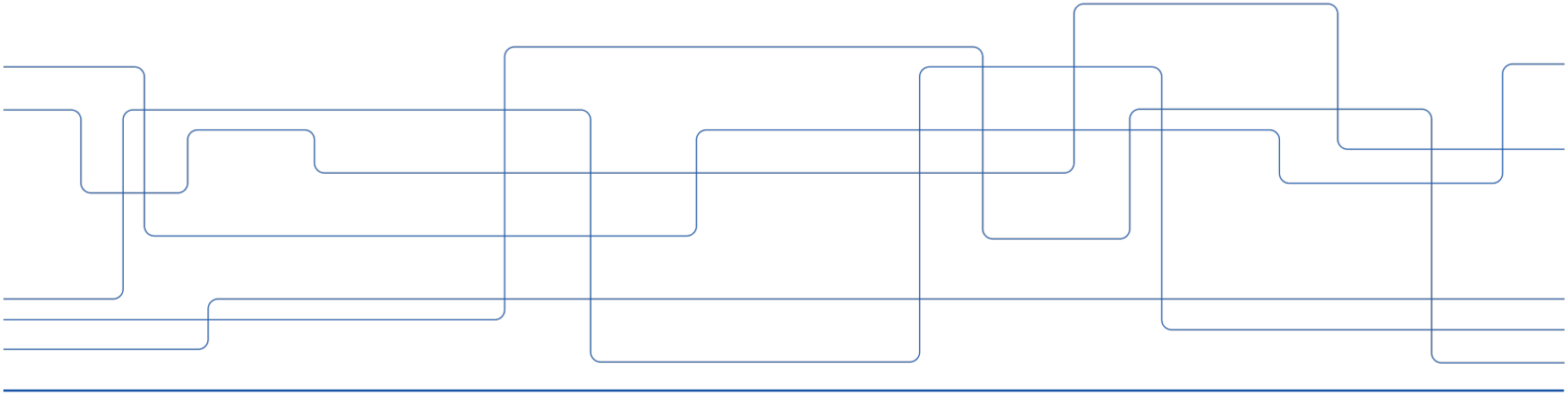
KTH ROYAL INSTITUTE
OF TECHNOLOGY

Using gem5 for Design Space Explorations

Bine Brank (FZJ), Dirk Pleiter (KTH)



Outline

- Methodological introduction
- Model construction and model parameter tuning
- Selected results
- Summary and conclusions

Methodological Introduction

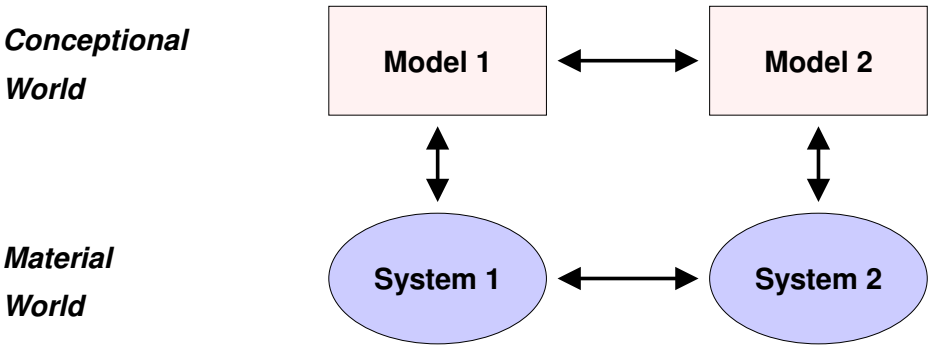
Definitions of the Term Model

- D. Hestenes (1986):
  - *“A model is a surrogate object, a conceptual representation of a real thing. The models in physics are mathematical models, which is to say that physical properties are represented by quantitative variables of the model”*
- B. Thalheim (2020):
  - *“A model is a well-formed, adequate, and dependable instrument that represents origins and that functions in utilisation scenarios”*
  - *“Models are used in various utilisation scenarios such as construction of systems, verification, optimization, explanation, and documentation”*

System versus Model

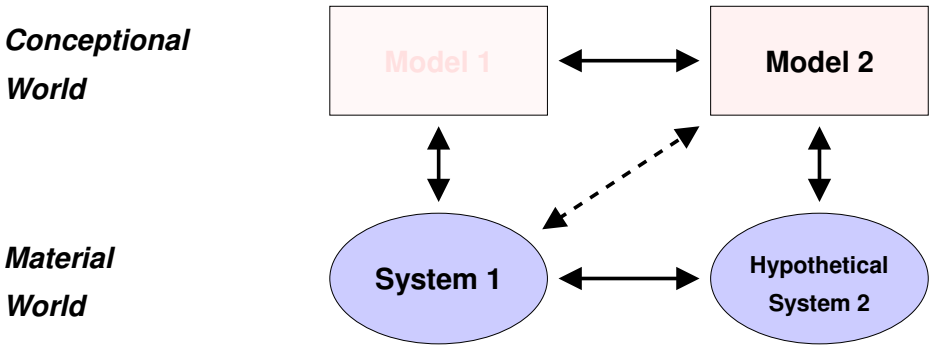
- Referential analogy: Empirical observations and measurements determine an analogy between a system and a model
- Conceptional analogy: Structural similarities between different models
- Material analogy: Relation of structures of different systems

[Hestenes, 2006]



Modified Approach

- Instead of creating a model for *System 1*, use empirical observations and measurements for a *Model 2* that is reasonable similar to *Model 1*
  - *Target scenario: Explore relation of architectural parameters and application performance*
- Interpretation: *Model 2* does not represent *System 1* (reference system), but a *Hypothetical System 2* that is analogue to *System 1*
- Introduce systematic changes in *Model 2* = **architectural exploration**



Methodology

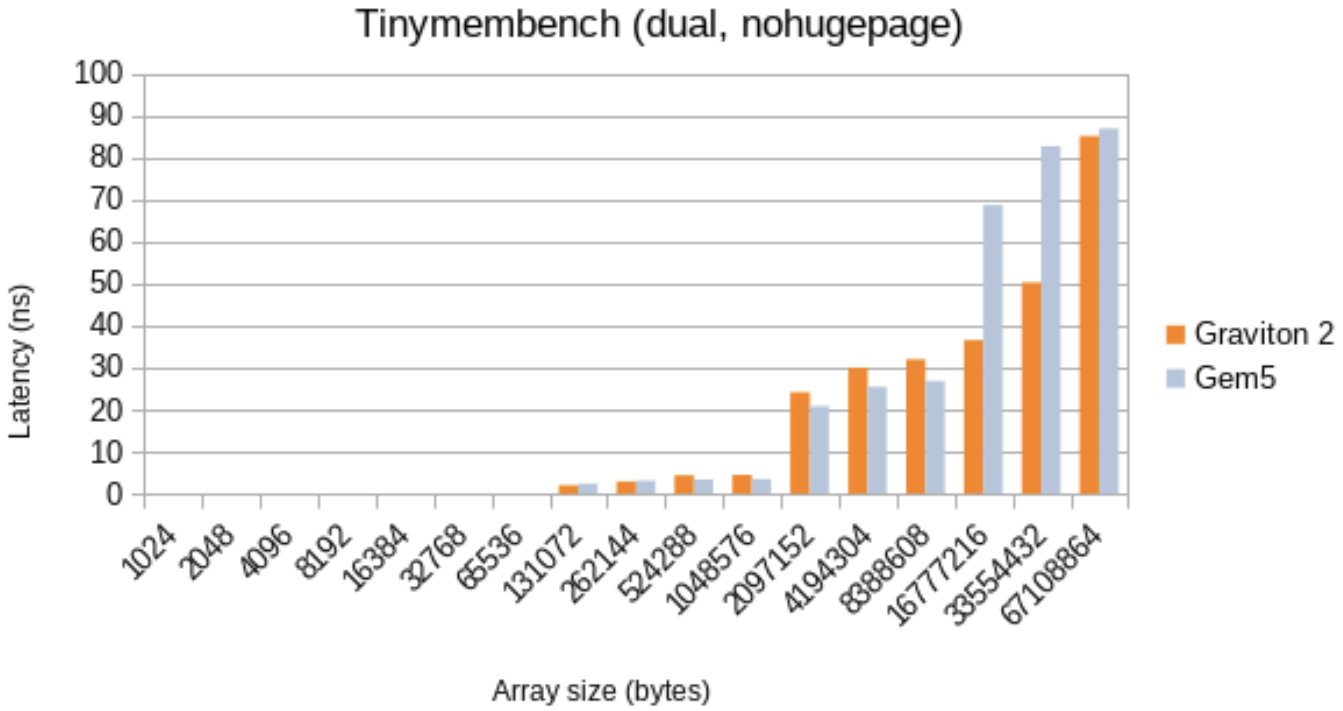
- Select reference architecture
- Model construction
  - *Configure gem5 model according to reference architecture selecting components represent the reference system well*
- Model parameter tuning
  - *Fix parameters according to technical specification or tune parameters based using micro-benchmarks*
    - Due to use of simplified sub-models for certain system components, effective parameter values need to be chosen
- Select configurations **and applications** for architecture exploration

Model Construction and Model Parameter Tuning

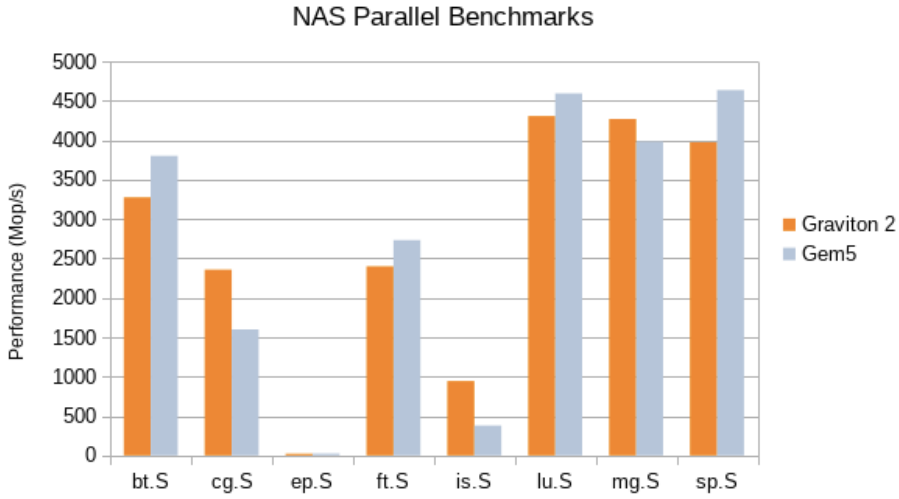
Implementation of the Methodology

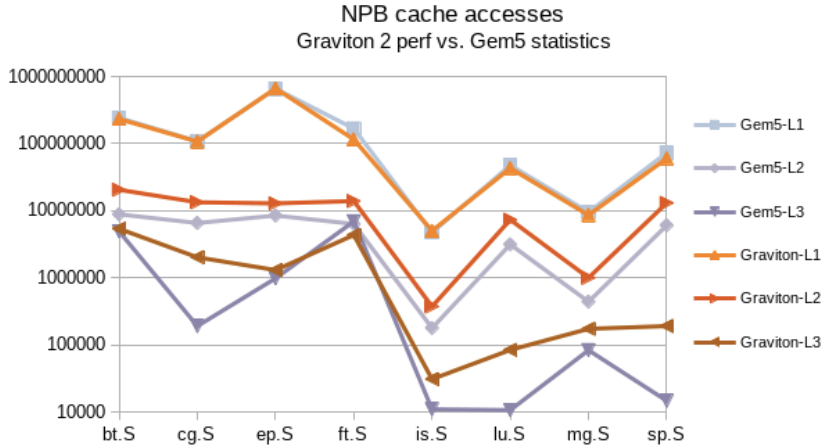
- Reference architecture: AWS Graviton 2
  - *Focus on a single core*
- Model construction: Choice of gem5 components
  - *O3CPU model, modified Arm_O3_v7 core*
  - *Classic memory system, tagged pre-fetcher at L2, L3 via crossbar*
- Micro-benchmarks
  - *STREAM, Tinymembench, NAS Parallel Benchmarks*
- Model modifications
  - *Support for SVE*

Model Parameter Tuning: Tinymembench



Model Parameter Tuning: NPB

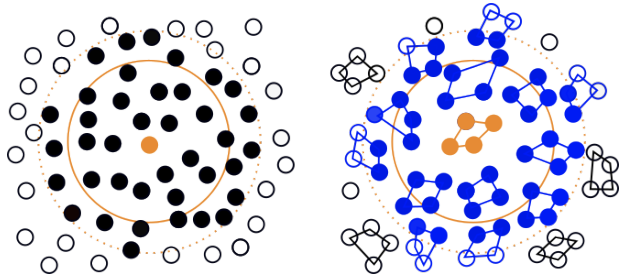




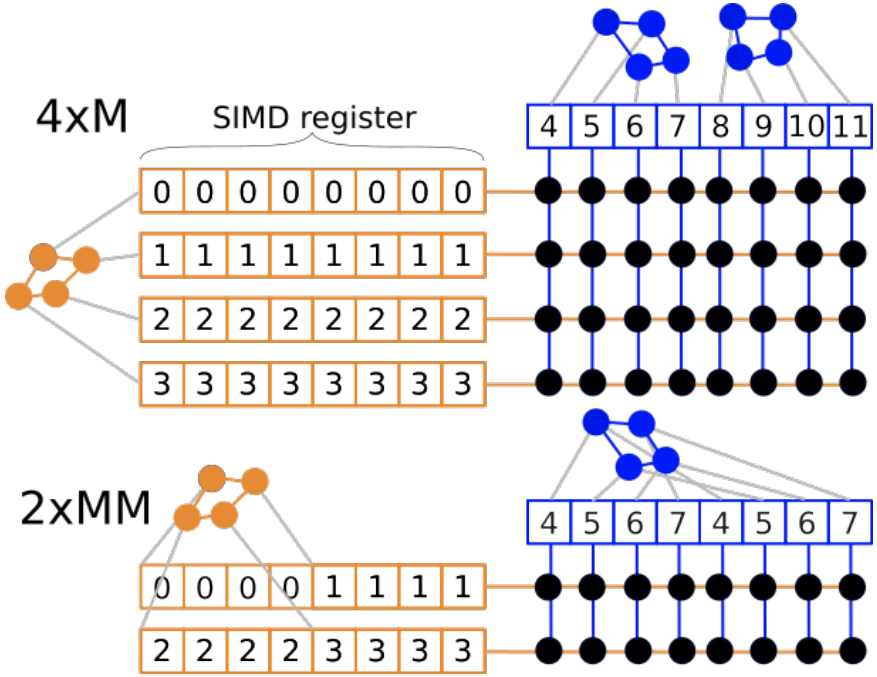
Selected Results

GROMACS: Force Calculation Non-bonded Atoms

- GROMACS implements a modified Verlet algorithm that is optimised for SIMD architectures

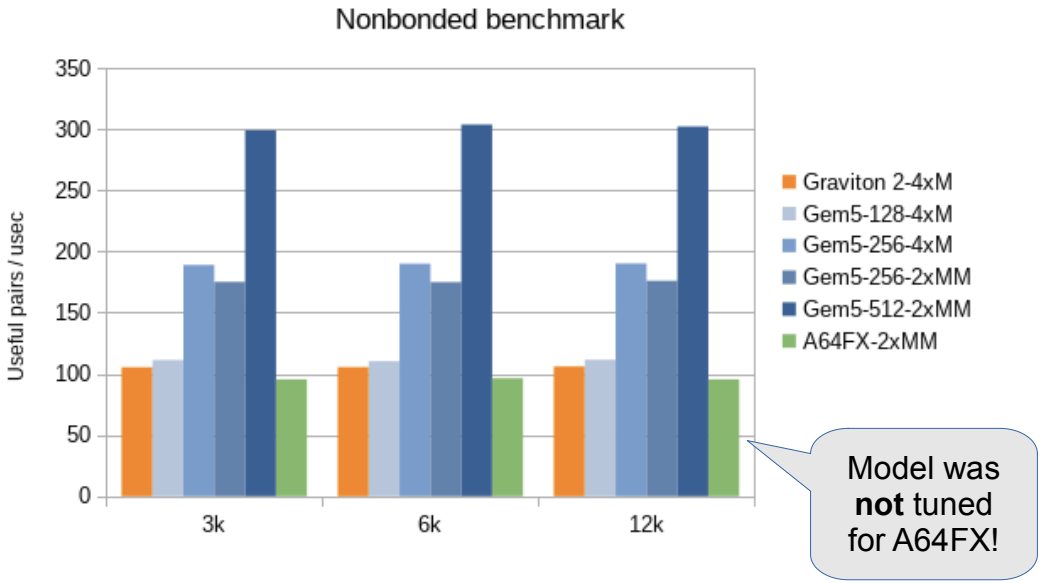


- Current SIMD-implementations:

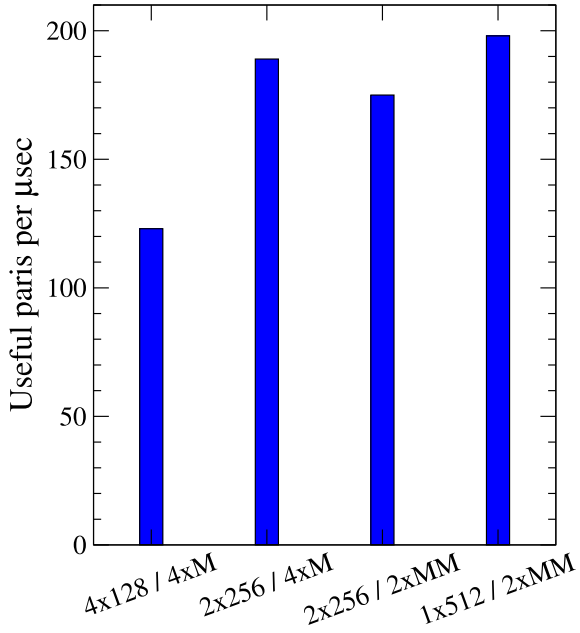


Number of SVE Pipelines and SVE Width

N_{SVE} = 2, variable b_{SVE} = 128, 256, 512



Constant N_{SVE} x b_{SVE} = 512



4xM versus 2xMM for N_{SVE} = 2

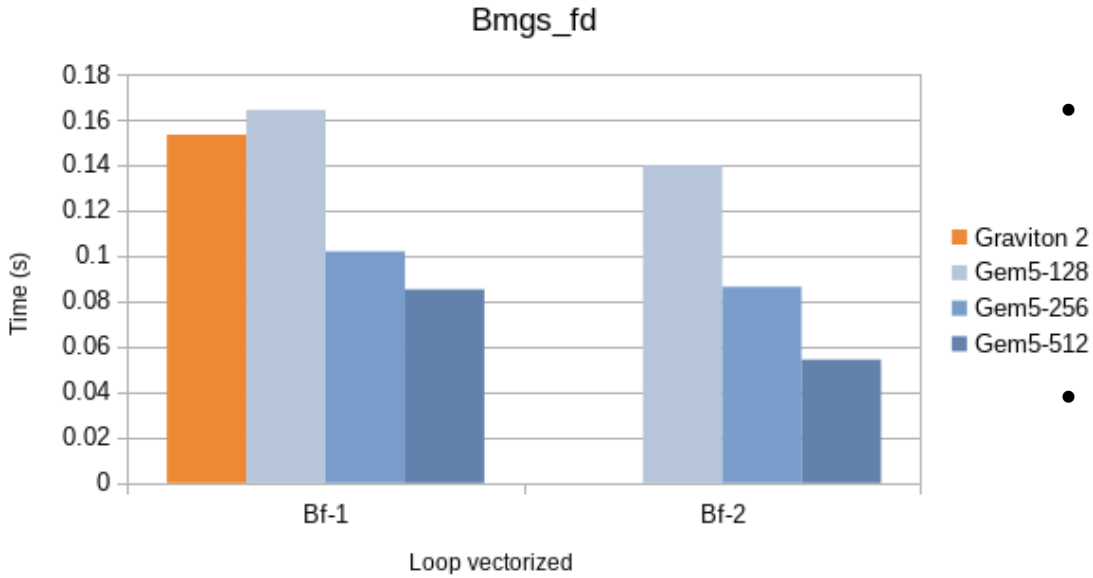
- Stalls due to full re-order buffer or full register files are performance relevant
  - *For fixed SVE width the number of full invents for 2xMM is larger compared to 2xM*
  - *For fixed implementation the number of full register events decreased for larger SVE width*
- Number of cache-line refills
  - *Increases for 2xMM compared 4xM*
  - *Decreases for larger SVE width*
    - Note increased capacity of the register file

GPAW

- Focus here: Discretised Laplace operator (`bmgs_fd` kernel)
  - *Memory-bound kernel*
- Compiler auto-vectorises inner-most loop
  - *Compiler generates gather load instructions*
- Outer-loop vectorisation needs to be done manually
  - *Use of intrinsics (including instructions for gather loads)*

```
 9 void
10 Z(bmgs_fd)(const bmgsstencil* s, const T* a, T* b)
11 {
12     /* Skip the leading halo area. */
13     a += (s->j[0] + s->j[1] + s->j[2]) / 2;
14
15     for (int i0 = 0; i0 < s->n[0]; i0++) {
16         for (int i1 = 0; i1 < s->n[1]; i1++) {
17 #ifdef _OPENMP
18 #pragma omp simd
19 #endif
20             for (int i2 = 0; i2 < s->n[2]; i2++) {
21                 int i = i2
22                       + i1 * (s->j[2] + s->n[2])
23                       + i0 * (s->j[1] + s->n[1] * (s->j[2] + s->n[2]));
24                 int j = i2 + i1 * s->n[2] + i0 * s->n[1] * s->n[2];
25                 T x = 0.0;
26
27                 for (int c = 0; c < s->ncoefs; c++)
28                     x += a[i + s->offsets[c]] * s->coefs[c];
29                 b[j] = x;
30             }
31         }
32     }
33 }
```

GPAW Performance



- Measured effective bandwidth far below STREAM performance
- Inner-loop vectorisation (Bf-1)
  - *Pipeline blocked by a non-pipelined horizontal reduction instruction*
- Outer-loop vectorisation (Bf-2)
  - *ld/st pipeline throughput limitation (74% busy for SVE-512)*

Summary and Conclusions

Summary and Conclusions

- Model theoretical review of our use of architecture simulations improves understanding of the results
- gem5 simulations provide information not only for hardware developer but, in particular, for application developers
  - *Feedback for application developers*
    - GROMACS
      - Good SIMD scaling: parallel efficiency of 68% comparing SVE-128 and SVE-512
      - Need for more ILP to exploit 4x SVE-128 pipelines
    - GPAW: Improve data-layout to reduce the number of ld/st micro-instructions
  - *Feedback for hardware developers*
    - GROMACS: Reduce stalls due to full ROB and/or register files
    - GPAW: Improve performance of scatter/gather instructions

Acknowledgements

Work has in part been performed on resources provided by the Open Edge and HPC Initiative

- OEHI provides access to a 30 node HPC cluster with Kunpeng 920 processors

http://www.openedgehpcinitiative.org/