
# COMPARATIVE ANALYSIS OF HPC APPLICATIONS ON ARM AND OTHER ARCHITECTURES

Dirk Pleiter, Stepan Nassyr | Arm HPC User Group, Dallas | 12.11.2018

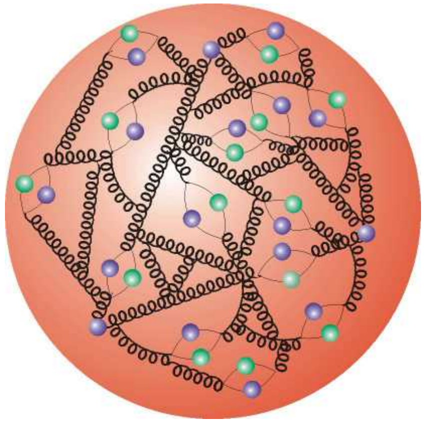
JÜLICH
Forschungszentrum

# Outline

- Methodology, applications and platforms
- Results
- Summary and conclusions

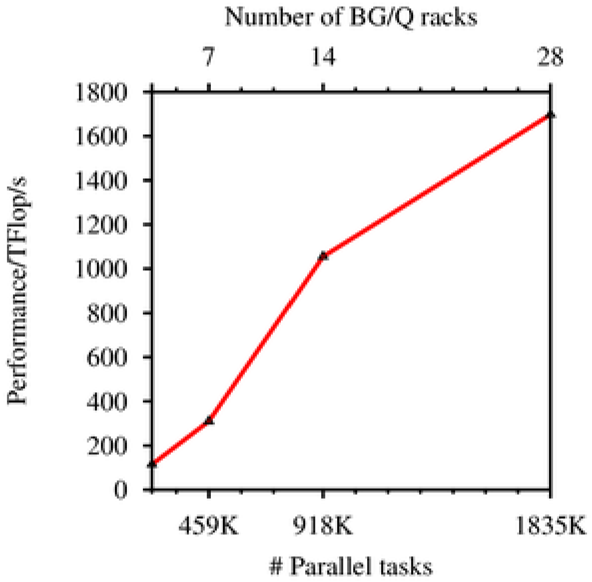
# METHODOLOGY, APPLICATIONS AND PLATFORMS

# Applications: Lattice QCD



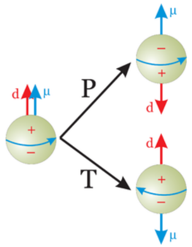
## Theory of strong interactions: Quantum Chromodynamics

- Quarks are the constituents of matter which strongly interact exchanging gluons
- Particular phenomena:
  - Confinement
  - Asymptotic freedom (Nobel Prize 2004)




## Selected science challenge: Upper bound for the neutron electric dipole moment

- Search for evidence for violations of charge and parity symmetry (CP symmetry) violations
- Piece in puzzle to understand the asymmetry of matter and anti-matter observed in the universe



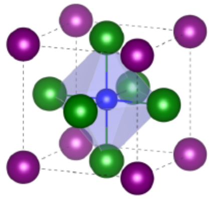

## Focus here: Iterative solvers for Wilson fermions

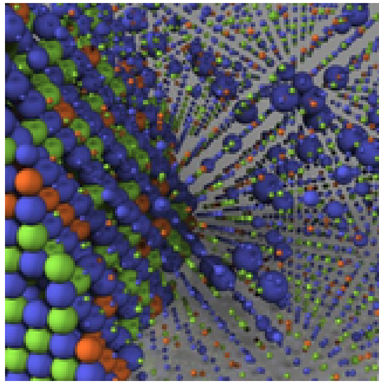
# Applications: Materials Sciences

## Density Functional Theory (DFT) for electronic structure calculations

- Method to compute electronic properties of a material
- Very widely used computational methods resulting in a huge number of publications

## Selected science challenges

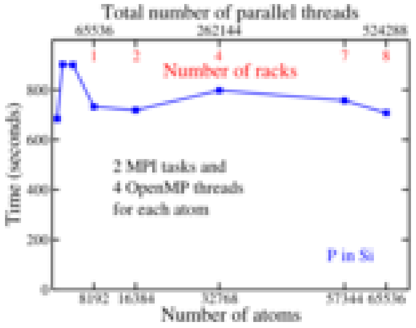


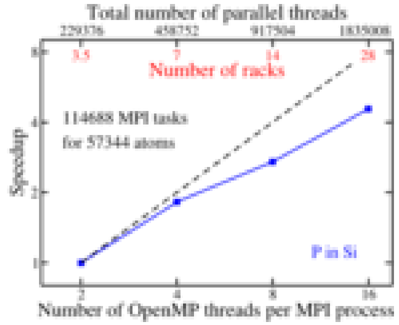


- Design Perovskite-based materials for photovoltaics
- Rational materials design for solid-state batteries
- Explore materials through simulations involving 100,000 atoms and more

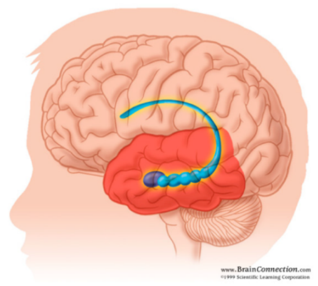
## Focus here

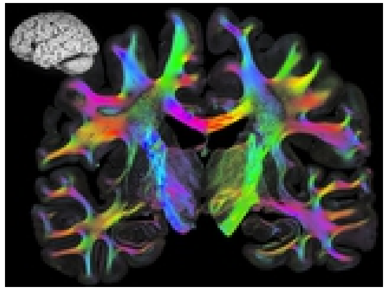
- KKRnano
- Quantum Espresso



# Applications: Brain Modelling



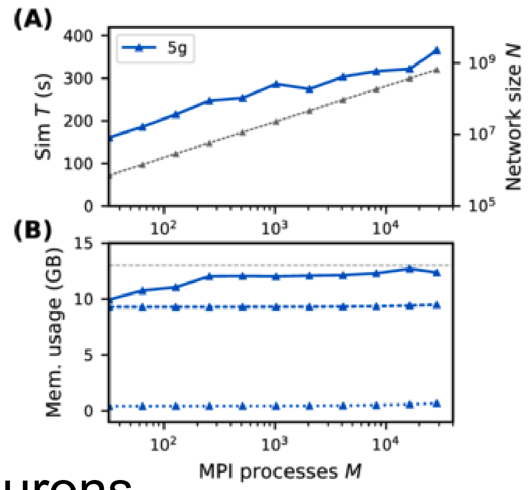


## Science challenges

- Create understanding of higher brain functions (learning, memory, spatial
- navigation) as well as dysfunctions causing mental diseases
- Create high-resolution atlases of the human brain
- Create biologically realistic models of the human brain

## Emerging field with a broad range of methods

- Advanced data analytics to process high-resolution images
- Simulation models with different levels of detail
  - Simulations using morphologically-detailed neurons
  - Spiking point-neurons for simulation of extremely large networks




## Focus here

- Neural simulation tools NEST for the case of spiking point-neurons

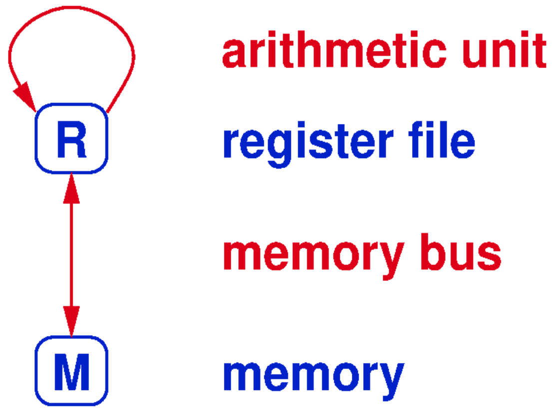
# Methodology

## Profiling and kernel identification

- Using HPCToolkit where necessary

## Kernel analysis

- Where possible determined Information Exchange function*:*
  - Amount of information $I$ that needs to be moved through moved through the system as function of the work-load $W$
  - Examples:
    - Amount of data loaded/stored from/to memory $I_{fp}(W)$
    - Number of floating-point operations $I_{mem}(W)$
- Modelling ansatz: $\Delta t = a_0 + a_1\ I(W)$




## Comparison of results

- Relate $(1/a_1)$ to hardware data transport or data processing capabilities

# Platforms

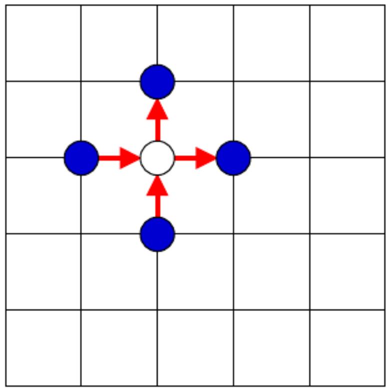
| | JULIA | JURON | JULIA-TX2 | JUAWEI |
|---|---|---|---|---|
| Processor | Broadwell | POWER8 | ThunderX2 (A2 stepping) | Hi1616 |
| #sockets * #cores | 2 * 14 | 2 * 10 | 2 * 28 | 2 * 32 |
| Clock frequency [GHz] | 2.4 | 3.5 | 2.0 | 2.0 |
| Nominal throughput of double-precision FP operations [GFlop/s] | 1075 | 560 | 896 | 1024 |
| Number of memory channels | 2 * 4 | 2 * 4 | 2 * 8 | 2 * 4 |
| Nominal memory bandwidth [GByte/s] | 154 | 230 | 341 | 136 |

# RESULTS

# Lattice QCD: Application Details

## Benchmark: Benchmark_wilson_sweep (Wilson even/odd)

- Part of C++ based Grid software toolkit [https://github.com/paboyle/Grid]
- Main numerical task within main kernel
- Parallelisation: SIMD, OpenMP, MPI
  - Port to NEON available [N. Meyer, 2018]



## Numerical task: sparse matrix-vector multiplication

- Matrix size for state-of-the-art calculations: $O(10^8)$

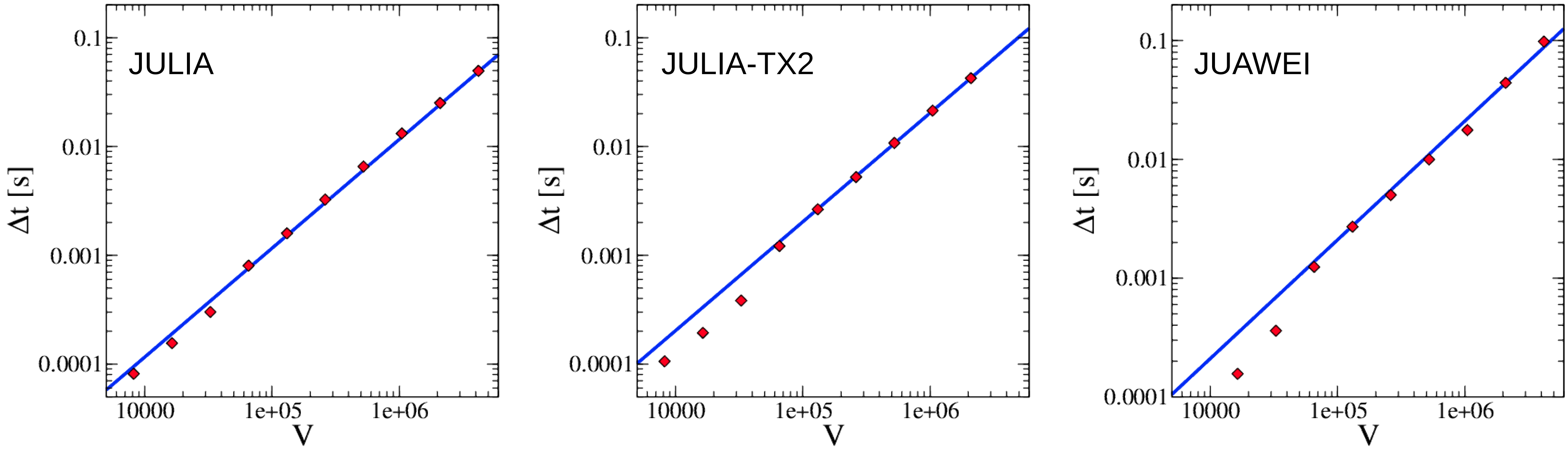
## Information exchange functions (double precision)

- $I_{\mathrm{fp}}(L,T) = (1320\ L^3\ T)$ Flop
- $I_{\mathrm{mem}}(L,T) = ((2688 + 192)\ L^3\ T)$ Byte
- $\mathrm{AI} = I_{\mathrm{fp}} / I_{\mathrm{mem}} = 0.46$ Flop/Byte



JÜLICH Forschungszentrum

# Lattice QCD: Performance Results




| | JULIA | JULIA-TX2 | JUAWEI |
|---|---|---|---|
| Optimal $N_{MPI}$ * $N_{thrd}$ | 16 * 1 | 16 * 2 | 8 * 8 |
| $b_{fp}$ [GFlop/s] | 56.9 | 32.5 | 31.4 |
| $B_{mem,eff}$ [GByte/s] | 124 | 71 | 69 |



JÜLICH Forschungszentrum

# Material Sciences: Applications Details and Profile

## miniKKR

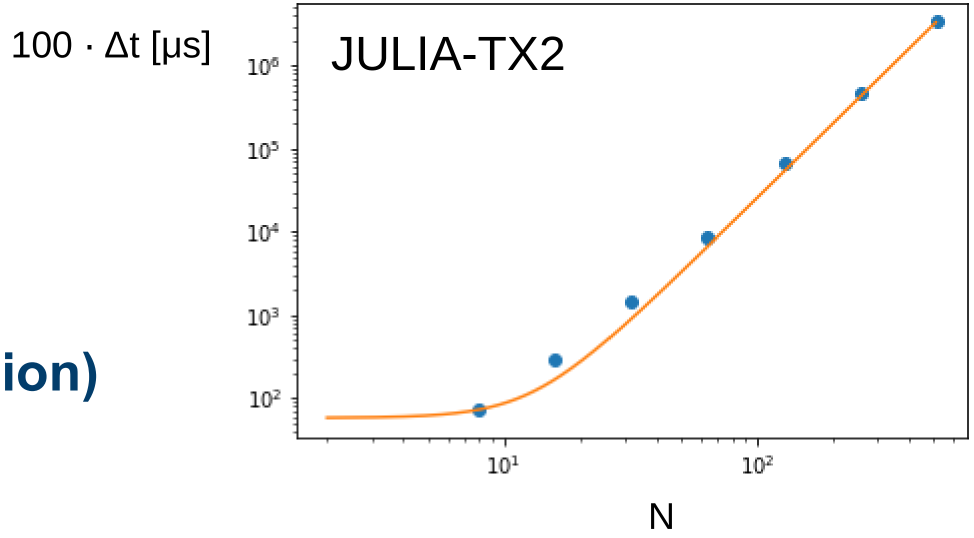
- Mini-application of KKRnano
- Main kernel: TFQMR iterative solver
  - Sparse-matrix multiplication
  - Matrix with substructure
    → implementation using zgemm()

| Scope | CPUTIME (usec):Sum (I) |
|---|---|
| Experiment Aggregate Metrics | 8.31e+07 100 % |
| <thread root> | 7.29e+07 87.7% |
| _INTERNAL_26______src_z_Linux_util_cpp_c3d2e46c::_k | 7.29e+07 87.7% |
| 585: __kmp_launch_thread | 7.29e+07 87.7% |
| 5810: __kmp_invoke_task_func | 5.02e+07 60.5% |
| 7212: __kmp_invoke_microtask | 5.02e+07 60.5% |
| outline bsrmm_mod.F90:240 (0x405baa) | 4.32e+07 51.9% |
| loop at bsrmm_mod.F90: 241 | 4.19e+07 50.5% |
| loop at bsrmm_mod.F90: 242 | 4.19e+07 50.5% |
| loop at bsrmm_mod.F90: 243 | 4.19e+07 50.5% |
| 271: zgemm_ | 4.19e+07 50.5% |

## Quantum Espresso (AUSURF112 benchmark)

- Main kernels (measurements on JULIA):
  - 28.3% zgemm
  - 23.2% FFT

# Material Sciences: OpenBLAS zgemm




## Numerical task

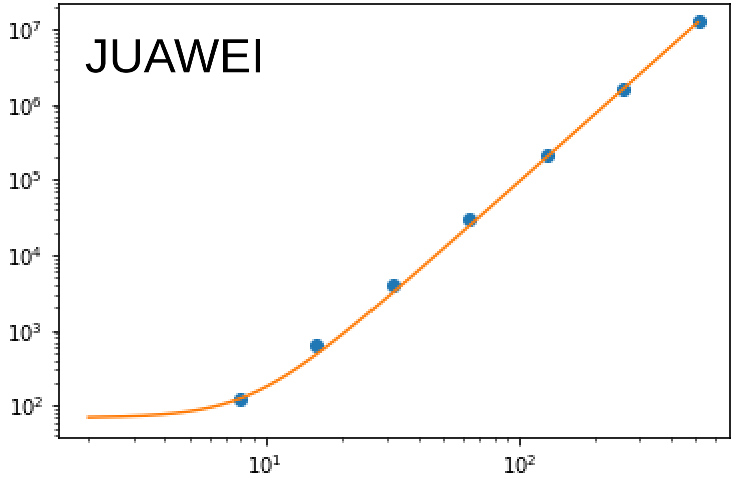
- $C \leftarrow \alpha \cdot \mathrm{op}(A) \cdot \mathrm{op}(B) + \beta \cdot \mathrm{op}(C)$

## Information exchange functions (double precision)

- $I_{fp}(N) = (8 \cdot N^3 + 12 \cdot N^2)$ Flop
- $I_{mem}(N) = (4 \cdot 16 \cdot N^2)$ Byte
- $AI = I_{fp} / I_{mem} = 0.125 \cdot N$ Flop/Byte (for large $N$)

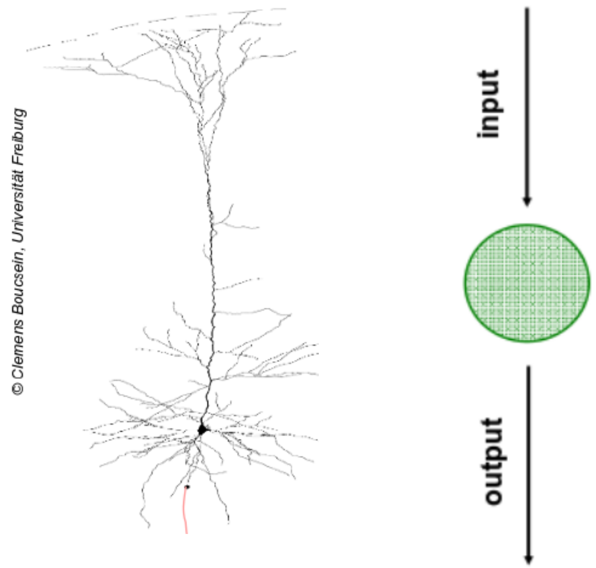
## Single core performance

| | x86 | JULIA-TX2 | JUAWEI |
|---|---|---|---|
| $b_{fp}$ [Flop/cycle] | 12.5 | 4 | 4 |




- Multi-thread performance performs badly on all platforms

# Brain Modelling: NEST Simulator

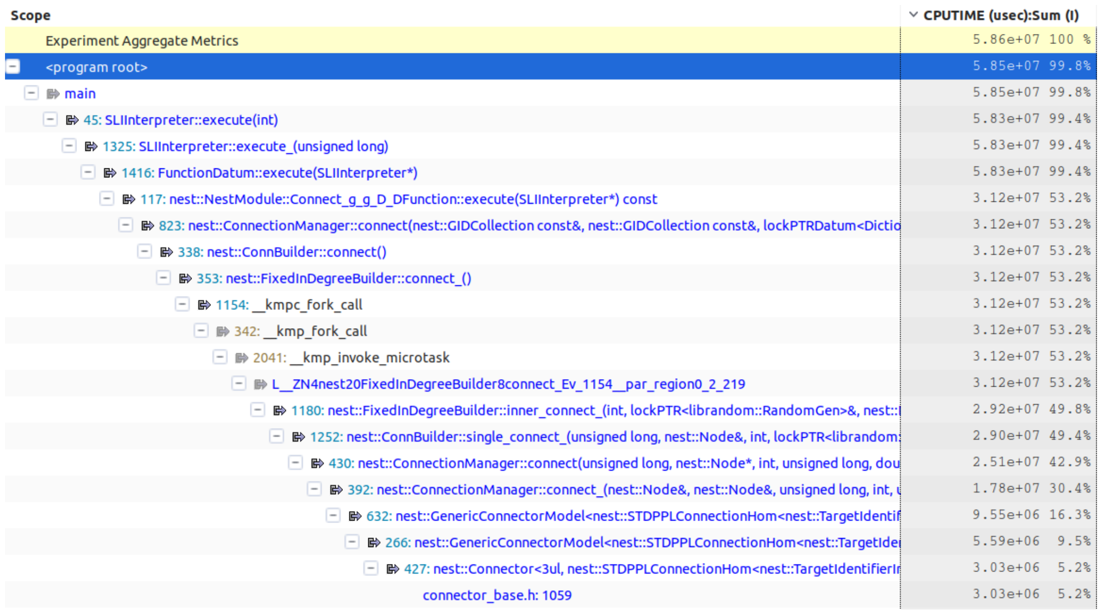



## Features

- Focus on simulation of point-neurons, simple integrate-and-fire model
- Application phases
  - Network construction and connection
  - Simulation

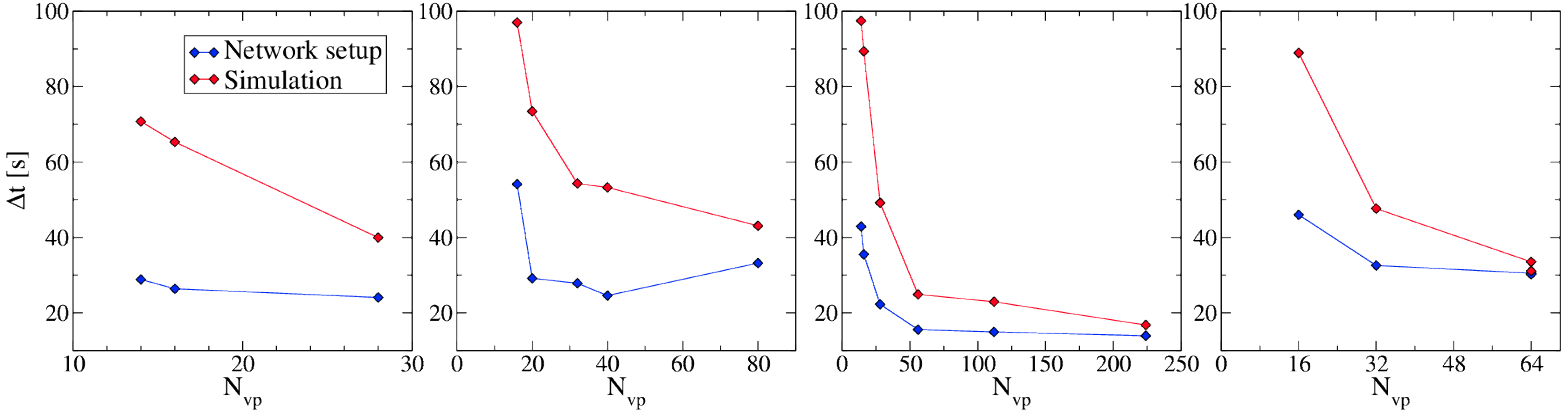
## Performance features

- High level of concurrency
- Simple models
  - Most time spent on spike delivery
  - Little time on neuron modelling
- Complex control flow



| Scope | CPUTIME (usec):Sum (I) |
|---|---|
| Experiment Aggregate Metrics | 5.86e+07 100 % |
| <program root> | 5.85e+07 99.8% |
| main | 5.85e+07 99.8% |
| 45: SLIInterpreter::execute(int) | 5.83e+07 99.4% |
| 1325: SLIInterpreter::execute_(unsigned long) | 5.83e+07 99.4% |
| 1416: FunctionDatum::execute(SLIInterpreter*) | 5.83e+07 99.4% |
| 117: nest::NestModule::Connect_g_g_D_DFunction::execute(SLIInterpreter*) const | 3.12e+07 53.2% |
| 823: nest::ConnectionManager::connect(nest::GIDCollection const&, nest::GIDCollection const&, lockPTRDatum<Dictio | 3.12e+07 53.2% |
| 338: nest::ConnBuilder::connect() | 3.12e+07 53.2% |
| 353: nest::FixedInDegreeBuilder::connect_() | 3.12e+07 53.2% |
| 1154: __kmpc_fork_call | 3.12e+07 53.2% |
| 342: __kmp_fork_call | 3.12e+07 53.2% |
| 2041: __kmp_invoke_microtask | 3.12e+07 53.2% |
| L__ZN4nest20FixedInDegreeBuilder8connect_Ev_1154__par_region0_2_219 | 3.12e+07 53.2% |
| 1180: nest::FixedInDegreeBuilder::inner_connect_(int, lockPTR<librandom::RandomGen>&, nest::l | 2.92e+07 49.8% |
| 1252: nest::ConnBuilder::single_connect_(unsigned long, nest::Node&, int, lockPTR<librandom: | 2.90e+07 49.4% |
| 430: nest::ConnectionManager::connect(unsigned long, nest::Node*, int, unsigned long, dou | 2.51e+07 42.9% |
| 392: nest::ConnectionManager::connect_(nest::Node&, nest::Node&, unsigned long, int, u | 1.78e+07 30.4% |
| 632: nest::GenericConnectorModel<nest::STDPPLConnectionHom<nest::TargetIdentif | 9.55e+06 16.3% |
| 266: nest::GenericConnectorModel<nest::STDPPLConnectionHom<nest::TargetIde | 5.59e+06 9.5% |
| 427: nest::Connector<3ul, nest::STDPPLConnectionHom<nest::TargetIdentifierI | 3.03e+06 5.2% |
| connector_base.h: 1059 | 3.03e+06 5.2% |

# Brain Modelling: NEST Performance Results



# Options for Improvement: Transcendental Functions

## Micro-benchmark

- Compute transcendental function for each element of a vector
- Use https://github.com/ARM-software/optimized-routines

## Results for exp() in cycles per element

| | gcc | XLC/ICC | Optimised library |
|---|---|---|---|
| JULIA | 59 | 9.4 | |
| JURON | 142 | 8.5 | |
| JULIA-TX2 | 68 | | 44 |

## Observations

- XLC and ICC perform vectorisation

# SUMMARY AND CONCLUSIONS

# Summary and Conclusions

## Review single node performance of vastly different applications

- LQCD: Structured Grids
- KKRnano / Quantum Espresso: Dense Linear Algebra
- NEST: Unstructured event simulation

## Comparison

- Bandwidth improvements cannot fully be exploited for LQCD
- Intel Xeon good for compute intensive applications
- ThunderX2 is outstanding for highly parallel, unstructured brain simulator

## Rooms for improvements

- Better understanding of options for improving memory access on ThunderX2
- Better support for complex linear algebra as well as transcendental functions