EXPLORING THE ARMV8 PROCESSOR ARCHITECTURE FOR HPC APPLICATIONS

18 September 2018 | Stepan Nassyr | Forschungszentrum Jülich

JÜLICH SUPERCOMPUTING CENTRE

Outline

Hardware at JSC

Software and Libraries used

- Performance tools
- Compilers, Libraries, Emulators

Applications

- KKRnano/MiniKKR
- Quantum Espresso
- NEST

Selected performance comparison

SVE status

Conclusions

Hardware at JSC

JSC production supercomputers

- JUWELS
  - 2511 x 2 Xeon Platinum 8168 (24 Cores, 2.7 Ghz)
  - 48 x (2 Xeon Gold 6148 (20 Cores, 2.0 Ghz) + 4x NVIDIA V100)
  - 2271 x 96 GiB, 240 x 192 GiB DDR4
- JURECA Cluster
  - 1872 x 2 Xeon E5-2680 v3 CPUs (12 Cores, 2.5 Ghz)
  - 75 nodes with 2x NVIDIA K80 GPU
  - 1605 x 128 GiB, 128 x 256 GiB, 64 x 512 GiB DDR4
- JURECA Booster
  - 1640 x Xeon Phi 7250-F (68 Cores, 1.4 Ghz)
  - 96 GiB DDR4 + 16 GiB MCDRAM
- QPACE 3
  - 672 Nodes with Intel Xeon Phi 7210 (64 Cores, 1.3 Ghz)

JSC prototypes

- JURON (IBM+NVIDIA)
  - 18 Nodes with
    - 2x10 IBM POWER8 Cores up to 4.023 Ghz
    - 4x NVIDIA Tesla P100
    - 256 GiB DDR4 + 4x16 GiB GPU HBM
- JULIA (CRAY)
  - 60 x Intel Xeon Phi 7210 (64 Cores, 1.3 Ghz)
  - 96 GiB DDR4 + 16 GiB MCDRAM per node

JSC ARM prototypes

- 4x Huawei Taishan 2160
  - Hi1610ES 32xARM Cortex-A57 @ 2.1 Ghz
  - 128 GiB DDR
- 12x Huawei Taishan 2180
  - Hi1612 32xARM Cortex-A57 @ 2.1 Ghz
  - 128 GiB DDR
- 12x Huawei Taishan 2280
  - Hi1616 32xARM Cortex-A72 @ >2.1 Ghz
  - 256 GiB DDR4
- 4x Cavium ThunderX2 Prototype
  - ThunderX2 CN9975-??? (prototype) 2x 28x ThunderX2 ARMv8.1-A cores @ 2.0 Ghz (empirically)
  - 4x SMT = 224 logical cores
  - 256 GiB DDR4

Software and Libraries used

HPCToolkit

- Used on Intel Xeon machines (v. 2018.08)
- Open source
- Performance counters support through PAPI and perf_event
- Statistical sampling
- Avoids instrumentation

ARM Map

- Used on ThunderX2 machines (v. 18.2.1)
- Developed and licenced by ARM as part of ARM Forge
- Performance counters support through PAPI
- Statistical sampling
- Avoids instrumentation
- More features (remote launch, compatible with multiple mpi implementations, ...)

Compilers and Libraries

- ARM HPC Compiler v. 18.4 for ARM machines
- Intel MKL, ICS, IMPI v. 2018.1.163 for Intel machines
- OpenMPI 3.1.2 on ARM
- ARM Performance Libraries v. 18.4.0

Emulation

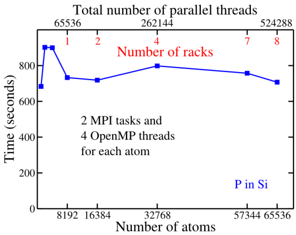
- ARM Instruction Emulator v. 18.2
- QEMU v. 3.0.0

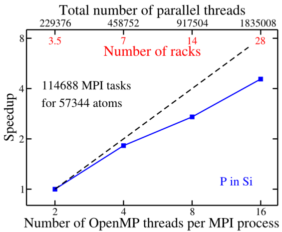
Applications

KKRnano

- DFT Electron Structure Code
- Written in Fortran 2003 + MPI, OpenMP
- linear-scaling
- Dominated by complex BLAS Kernels
- Mini-APP miniKKR: small (8x8,16x16,32x32) ZGEMMs
- Part of High-Q club

KKRnano: High-Q





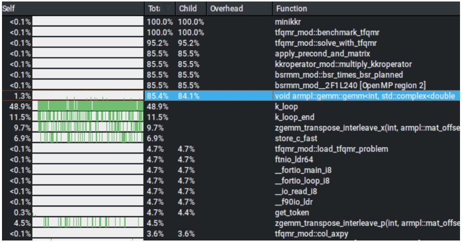
(left) weak-scaling obtained by increasing the number of MPI tasks together with the number of atoms, (right) scaling to the full machine by increasing the number of OpenMP threads while keeping the number of MPI tasks fixed. Both use 64 hardware threads on each node. (Taken from KKRnano High-Q results)

MiniKKR: Profile

Scope	CPUTIME (usec):Sum (I)	
outline bsrmm_mod.F90:240 (0x405baa)	3.85e+07	73.8%
zgemm_	3.85e+07	73.7%
mkl_blas_zgemm	3.83e+07	73.4%
mkl_blas_zgemm_host	3.83e+07	73.3%
mkl_blas_avx2_xzgemm	3.78e+07	72.5%
mkl_blas_avx2_z_generic_fullacopybcopy	3.75e+07	71.9%
mkl_blas_avx2_zgemm_ker0	2.58e+07	49.5%
mkl_blas_avx2_zgemm_kernel_0	2.35e+07	45.0%
mkl_blas_avx2_zgemm_copyan	7.09e+06	13.6%
mkl_blas_avx2_zgemm_zcopy_down6_ea	7.05e+06	13.5%
mkl_blas_avx2_zgemm_copybn	2.72e+06	5.2%
mkl_blas_avx2_zgemm_zcopy_right2_ea	2.70e+06	5.2%
mkl_blas_avx2_zgemm_kernel_0_b0	2.27e+06	4.3%
outline tfQMR_mod.F90:574 (0x415368)	2.23e+06	4.3%
tfqmr_mod_mp_load_tfqmr_problem_	2.23e+06	4.3%
for_read_seq_lis	2.23e+06	4.3%
for_read_seq_lis_xmit	2.23e+06	4.3%
tfqmr_mod_mp_col_axpy_	2.23e+06	4.3%

Broadwell profile generated with HPCToolkit v. 2018.08

MiniKKR: Profile



Self	Tot:	Child	Overhead	Function
<0.1%	100.0%	100.0%		minikkr
<0.1%	100.0%	100.0%		tfqmr_mod::benchmark_tfqmr
<0.1%	95.2%	95.2%		tfqmr_mod::solve_with_tfqmr
<0.1%	85.5%	85.5%		apply_precond_and_matrix
<0.1%	85.5%	85.5%		kkroperator_mod::multiply_kkroperator
<0.1%	85.5%	85.5%		bsrmm_mod::bsr_times_bsr_planned
<0.1%	85.5%	85.5%		bsrmm_mod__2F1L240 [OpenMP region 2]
1.3%	85.4%	84.1%		void armpl::gemm::gemm<int, std::complex<double
48.9%	48.9%			k_loop
11.5%	11.5%			k_loop_end
9.7%	9.7%			zgemm_transpose_interleave_x(int, armpl::mat_offse
6.9%	6.9%			store_c_fast
<0.1%	4.7%	4.7%		tfqmr_mod::load_tfqmr_problem
<0.1%	4.7%	4.7%		ftnio_ldr64
<0.1%	4.7%	4.7%		_fortio_main_i8
<0.1%	4.7%	4.7%		_fortio_loop_i8
<0.1%	4.7%	4.7%		_io_read_i8
<0.1%	4.7%	4.7%		_f90io_ldr
0.3%	4.7%	4.4%		get_token
4.5%	4.5%			zgemm_transpose_interleave_p(int, armpl::mat_offse
<0.1%	3.6%	3.6%		tfqmr_mod::col_axpy

ThunderX2 profile generated with ARM Forge 18.2.1

MiniKKR: Profile

- profile shown is for 1 process and 1 thread
- ZGEMM dominates, next is ZAXPY with < 5%
- For ThunderX2 loading the problem actually starts to dominate due to shorter compute time (slow file system)
- time spent in _kmp_barrier a lot higher for the Intel machine (might be hidden, <unknown> in libarmpl_lp64.so shows up in profile)

ZGEMM and sync CPU time with increasing thread count:

# Threads	ZGEMM Xeon	ZGEMM ThunderX2	sync Xeon	sync ThunderX2
1	73.7%	85.4%	0%	0%
2	69.5%	78.0%	14.8%	1.1%
4	63.2%	70.4%	23.0%	2.7%
8	57.3%	59.1%	30.0%	3.8%

MiniKKR: Properties of major kernels

- ZGEMM:
  - Block-sparse MM
  - Dense MM of small blocks (8x8,16x16,32x32)
- ZAXPY, ZXPAY, ZDOTU:
  - contribution < 5 %
  - Applied to blocks-size vectors
  - Memory-Bandwidth-limited

Quantum Espresso

- Electron structure code
- Fortran + MPI
- Complex arithmetic
- Dominated by large (5000 < n < 6000) ZGEMMs

Quantum Espresso: Profile

Scope	CPUTIME (usec):Sum (I)	
run_pwscf_	3.10e+10	100.0
electrons_	3.01e+10	96.9%
electrons_scf_	3.01e+10	96.9%
c_bands_	2.68e+10	86.4%
diag_bands_	2.68e+10	86.3%
diag_bands_IP_diag_bands_k_	2.68e+10	86.3%
cegterg_	2.66e+10	85.9%
cdiaghg_	1.13e+10	36.3%
pmpi_barrier_	1.11e+10	35.7%
MPIR_Barrier	1.11e+10	35.7%
I_MPI_COLL_SHM_GENERIC_RELEASE_BCAST..0	1.10e+10	35.6%
I_MPI_COLL_SHM_FLAT_RELEASE	1.10e+10	35.6%
[I] mp_synchronize	1.10e+10	35.6%
bcast_real_	1.09e+10	35.0%
mp_mp_mp_bcast_rv_	1.08e+10	34.8%
h_psi_	9.41e+09	30.3%
h_psi__	9.41e+09	30.3%
mkl_blas_avx2_z_generic_fullacopybcopy	8.85e+09	28.5%
mkl_blas_zgemm	8.84e+09	28.5%
mkl_blas_zgemm_host	8.84e+09	28.5%
mkl_blas_avx2_xzgemm	8.84e+09	28.5%
zgemm_	8.79e+09	28.3%
mkl_blas_avx2_zgemm_ker0	8.42e+09	27.1%

Broadwell profile generated with HPCToolkit v. 2018.08

Quantum Espresso: Profile

Self	Tot:	MPI	Child	Function
<0.1%	93.8%	58.4%	93.8%	pwscf
<0.1%	93.8%	58.4%	93.8%	run_pwscf
<0.1%	92.8%	58.2%	92.8%	electrons
<0.1%	92.8%	58.2%	92.8%	electrons_scf
<0.1%	88.6%	57.8%	88.6%	c_bands
<0.1%	88.5%	57.8%	88.5%	diag_bands
<0.1%	88.5%	57.8%	88.5%	diag_bands_k [inlined]
0.1%	88.1%	57.8%	88.0%	cegterg
<0.1%	58.1%	56.1%	58.1%	cdiaghg
<0.1%	56.8%	56.8%	56.8%	mp_synchronize [inlined]
56.8%	56.8%	56.8%		mpi_barrier_
<0.1%	56.1%	56.1%	56.1%	bcast_real
<0.1%	56.0%	56.0%	56.0%	mp::mp_bcast_rv
<0.1%	28.0%		28.0%	void armpl::gemm::gemm<int, std::complex<double

ThunderX2 profile generated with ARM Forge 18.2.1

Quantum Espresso: Profile

- profile for 28 processes and nt=4 (4 task groups with 7 processes each)
- 56.8% vs 35.7% (ThunderX2 vs Broadwell) CPU time spent on MPI synchronization
- 26.7% vs 28.5% (ThunderX2 vs Broadwell) CPU time spent on ZGEMM
- Other kernels < 5% on both

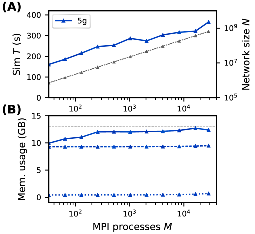
Quantum Espresso: Properties of ZGEMM

- ZGEMM:
    - Larger dense MM
    - $O(N^3)$ with matrix size
    - High arithmetic intensity

NEST

- Simulator for spiking neurons
- Modern C++ code
- Also part of the High-Q club
- Memory requirements per node rise with more nodes

NEST: High-Q



Weak scaling of an archetypal neural network simulation. Runtime and memory usage per compute node with 1 MPI process and 64 threads per node. Each compute node hosts 22,500 neurons with 11,250 incoming synapses per neuron. (A) Simulation time. Gray triangles and dashed line show the total network size N (right vertical axis). (B) Cumulative memory usage for a single MPI process after construction of neurons (dotted; < 140 MB), after construction of connections (dashed), and after simulation (solid). (Taken from NEST High-Q results)

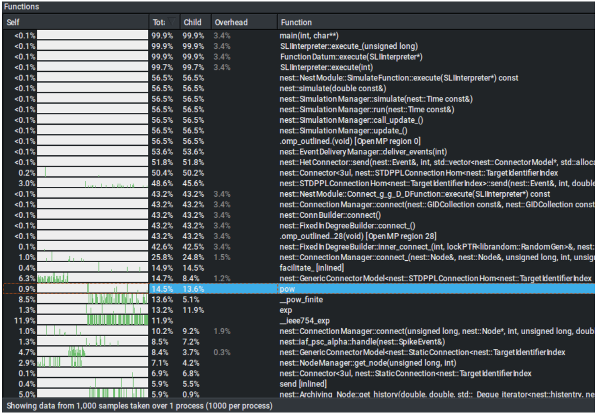
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

NEST: Profile

Scope	CPUTIME (usec):Sum (I)
nest::EventDeliveryManager::deliver_events(int)	2.51e+07 42.8%
nest::HetConnector::send(nest::Event&, int, std::vector<nest::ConnectorM	2.33e+07 39.7%
nest::ConnectionManager::connect_(nest::Node&, nest::Node&, unsigned	1.79e+07 30.5%
nest::STDPPLConnectionHom<nest::TargetIdentifierIndex>::send(nest::Eve	1.71e+07 29.1%
nest::GenericConnectorModel<nest::STDPPLConnectionHom<nest::Target	9.59e+06 16.4%
nest::GenericConnectorModel<nest::StaticConnection<nest::TargetIdentifi	5.79e+06 9.9%
nest::GenericConnectorModel<nest::STDPPLConnectionHom<nest::Target	5.59e+06 9.5%
nest::NodeManager::get_node(unsigned long, int)	5.48e+06 9.3%
nest::RingBuffer::add_value(long, double)	5.26e+06 9.0%
nest::GenericConnectorModel<nest::StaticConnection<nest::TargetIdentifi	4.41e+06 7.5%
nest::StaticConnection<nest::TargetIdentifierIndex>::send(nest::Event&, in	4.15e+06 7.1%
nest::Connector<3ul, nest::STDPPLConnectionHom<nest::TargetIdentifierI	4.11e+06 7.0%
__libm_exp_l9	3.86e+06 6.6%
nest::STDPPLConnectionHom<nest::TargetIdentifierIndex>::facilitate_(dou	3.48e+06 5.9%
nest::VPManager::get_thread_id() const	3.29e+06 5.6%
__kmp_api_omp_get_thread_num	3.24e+06 5.5%
__libm_pow_l9	3.22e+06 5.5%
nest::Archiving_Node::get_history(double, double, std::_Deque_iterator<ne	3.12e+06 5.3%
nest::Connection<nest::TargetIdentifierIndex>::check_connection_(nest::N	2.82e+06 4.8%
update_get_addr	2.62e+06 4.5%
_dl_update_slotinfo	2.45e+06 4.2%
nest::Archiving_Node::get_K_value(double)	2.16e+06 3.7%
std::vector<nest::SparseNodeArray::NodeEntry, std::allocator<nest::Spars	1.94e+06 3.3%

Broadwell profile generated with HPCToolkit v. 2018.08

NEST: Profile



Functions

Self	Total	Child	Overhead	Function
<0.1%	99.9%	99.9%	3.4%	main(int, char**)
<0.1%	99.9%	99.9%	3.4%	SLIInterpreter::execute_(unsigned long)
<0.1%	99.9%	99.9%	3.4%	FunctionDatum::execute(SLIInterpreter*)
<0.1%	99.7%	99.7%	3.4%	SLIInterpreter::execute(int)
<0.1%	56.5%	56.5%		nest::NestModule::SimulateFunction::execute(SLIInterpreter*) const
<0.1%	56.5%	56.5%		nest::simulate(double const&)
<0.1%	56.5%	56.5%		nest::SimulationManager::simulate(nest::Time const&)
<0.1%	56.5%	56.5%		nest::SimulationManager::run(nest::Time const&)
<0.1%	56.5%	56.5%		nest::SimulationManager::call_update_()
<0.1%	56.5%	56.5%		nest::SimulationManager::update_()
<0.1%	56.5%	56.5%		.omp_outlined.(void) [OpenMP region 0]
<0.1%	53.6%	53.6%		nest::EventDeliveryManager::deliver_events(int)
<0.1%	51.8%	51.8%		nest::HetConnector::send(nest::Event&, int, std::vector<nest::ConnectorModel*, std::alloca
0.2%	50.4%	50.2%		nest::Connector<3ul, nest::STDPPLConnectionHom<nest::TargetIdentifierIndex
3.0%	48.6%	45.6%		nest::STDPPLConnectionHom<nest::TargetIdentifierIndex>::send(nest::Event&, int, double
<0.1%	43.2%	43.2%	3.4%	nest::NestModule::Connect_g_g_D_DFunction::execute(SLIInterpreter*) const
<0.1%	43.2%	43.2%	3.4%	nest::ConnectionManager::connect(nest::GIDCollection const&, nest::GIDCollection const
<0.1%	43.2%	43.2%	3.4%	nest::ConnBuilder::connect()
<0.1%	43.2%	43.2%	3.4%	nest::FixedInDegreeBuilder::connect_()
<0.1%	43.2%	43.2%	3.4%	.omp_outlined..28(void) [OpenMP region 28]
0.1%	42.6%	42.5%	3.4%	nest::FixedInDegreeBuilder::inner_connect_(int, lockPTR<librandom::RandomGen>&, nest:
1.0%	25.8%	24.8%	1.5%	nest::ConnectionManager::connect_(nest::Node&, nest::Node&, unsigned long, int, unsig
0.4%	14.9%	14.5%		facilitate_ [inlined]
6.3%	14.7%	8.4%	1.2%	nest::GenericConnectorModel<nest::STDPPLConnectionHom<nest::TargetIdentifierIndex
0.9%	14.5%	13.6%		pow
8.5%	13.6%	5.1%		__pow_finite
1.3%	13.2%	11.9%		exp
11.9%	11.9%			__ieee754_exp
1.0%	10.2%	9.2%	1.9%	nest::ConnectionManager::connect(unsigned long, nest::Node*, int, unsigned long, doub
1.3%	8.5%	7.2%		nest::iaf_psc_alpha::handle(nest::SpikeEvent&)
4.7%	8.4%	3.7%	0.3%	nest::GenericConnectorModel<nest::StaticConnection<nest::TargetIdentifierIndex
2.9%	7.1%	4.2%		nest::NodeManager::get_node(unsigned long, int)
0.1%	6.9%	6.8%		nest::Connector<3ul, nest::StaticConnection<nest::TargetIdentifierIndex
0.4%	5.9%	5.5%		send [inlined]
5.0%	5.9%	0.9%		nest::Archiving_Node::get_history(double, double, std::_Deque_iterator<nest::histentry, ne

Showing data from 1,000 samples taken over 1 process (1000 per process)

ThunderX2 profile generated with ARM Forge 18.2.1

NEST: Profile

- Profile for 1 process and 1 thread.
- 52.8% vs 53.2% (ThunderX2 vs Broadwell) CPU time spent on deliver_events
- 42.1% vs 42.8% (ThunderX2 vs Broadwell) CPU time spent on connect()/initialization
- ”Simple” computations in loops:
    - 13.0% vs 6.6% spent on exp()
    - 13.8% vs 5.5% spent on pow()
- Roughly double of CPU time spent on pow() and exp() on ThunderX2
- No single dominant ”kernel”

Selected performance comparison

Quantum Espresso: ThunderX2 vs Broadwell

- QE version 6.2.1
- AUSURF112 input data

# Processes	# nt	Xeon E5 2680 v4	Xeon cpufreq	ThunderX2
1	1	165m (22 iter)	3.3 Ghz	673m (22 iter)
14	2	22m 14s (22 iter)	2.9 Ghz	84m 33s (22 iter)
28	4	18m 25s (22 iter)	2.9 Ghz	59m 30s (22 iter)
28	7	21m 32s (25 iter)	2.9 Ghz	67m 38s (25 iter)
56	14	-	-	51m 13s (22 iter)

- 2x AVX2 FPU = 16 dflops/cycle, 2xNEON FPU = 8 dflops/cycle
- Xeon boosts up to 2.9 Ghz, CN9975 @ 2 Ghz
- Profile: lower MPI cost on Broadwell

MiniKKR: ThunderX2 vs Broadwell

- Current git version

# Threads	Problem Set	Xeon E5 2680 v4	ThunderX2	Quotient
1	A	46.2s	191.1s	4.13
2	A	27.8s	105s	3.78
4	A	16.4s	56.9	3.47
8	A	9.6s	33.2	3.46
14	B	95s	309s	3.25
28	B	69s	206s	2.98
56	B	-	158s	-

NEST: ThunderX2 vs Broadwell

# Processes	# Threads/Process	# Neurons	Xeon E5 2680 v4	ThunderX2
1	1	11250	59 s	181 s
7	2	157500	136 s	248 s
14	1	157500	126 s	253 s
7	4	157500	87 s	144 s
14	2	157500	63 s	137 s
28	1	157500	68 s	135 s
14	4	157500	76 s	73 s
28	2	157500	64 s	71 s
56	1	157500	62 s	68 s
28	4	315000	151 s	129 s
112	1	315000	-	126 s

NEST: ThunderX2 vs Broadwell

- NEST v. 2.14

- Single thread: Broadwell faster by factor 3.06 (1.86 when accounting for clock speed)
- Only uses max. 4 cores per process on ThunderX2
- Scales really well with higher number of threads/processes on ARMv8
- Lack of dense kernels takes AVX advantage

SVE status

Exploitation Opportunities

- Auto-Vectorization
- BLAS kernels
- Complex arithmetic

Status of porting and verification

- MiniKKR, NEST, QE - SVE compilation successful with:
  - ARM HPC Compiler v. 18.4
  - GCC v. 8.2.0
- Custom SVE kernels for MiniKKR with ACLE (ZGEMM, ZAXPY, ZXPAY, ZDOTU)
- Emulation with ARMIE:
  - Correct results when single threaded (MiniKKR, NEST)
  - Issues with OpenMP
- Emulation with QEMU:
  - Works perfectly with GCC v. 8.2.0 compiled binaries
  - Issues with OpenMP when using ARM HPC Compiler
- Emulation with GEM5:
  - Currently work in Progress
  - First successes with simple programs

Conclusions

Conclusions

- Current ARMv8 - based systems show promising performance
- Functional SVE emulation possible
- Expectation of higher performance with SVE especially in BLAS-heavy applications