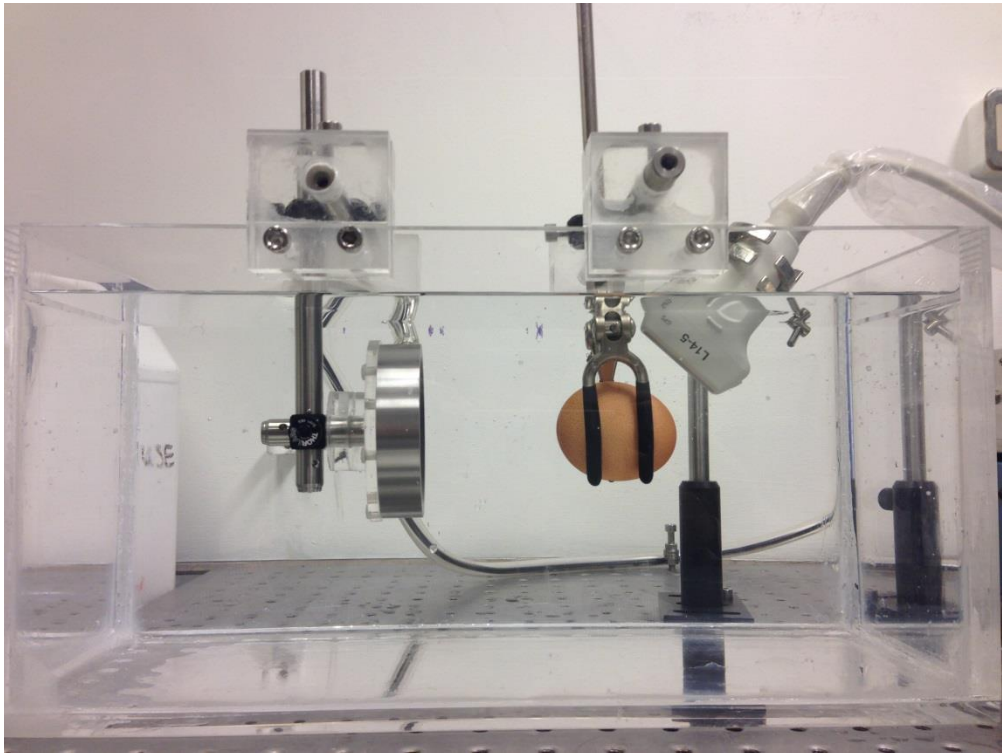
Acceleration of the k-Wave toolbox on Xeon Phi

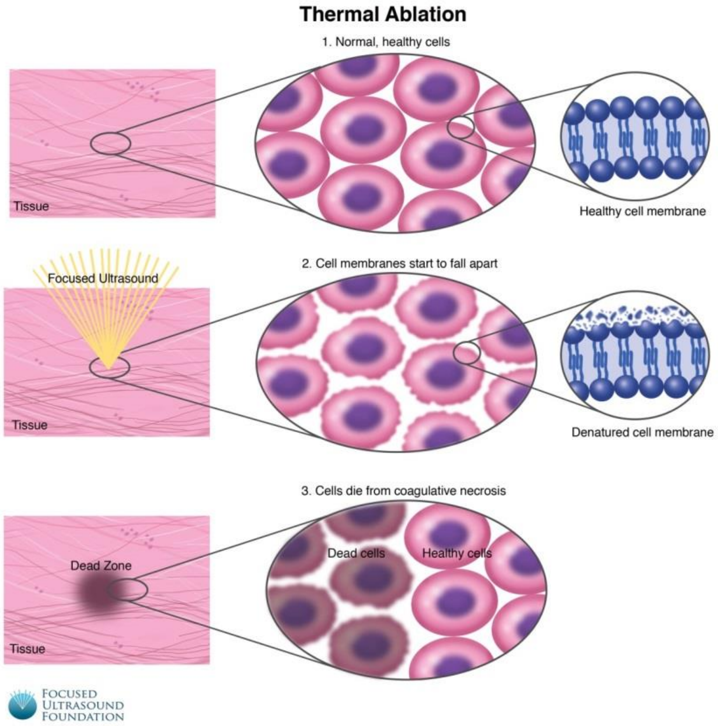
Jiri Jaros

Supercomputing Technologies Research Group
Faculty of Information Technology, Brno University of Technology
Božetěchova 2, 612 66 Brno
jarosjir@fit.vutbr.cz

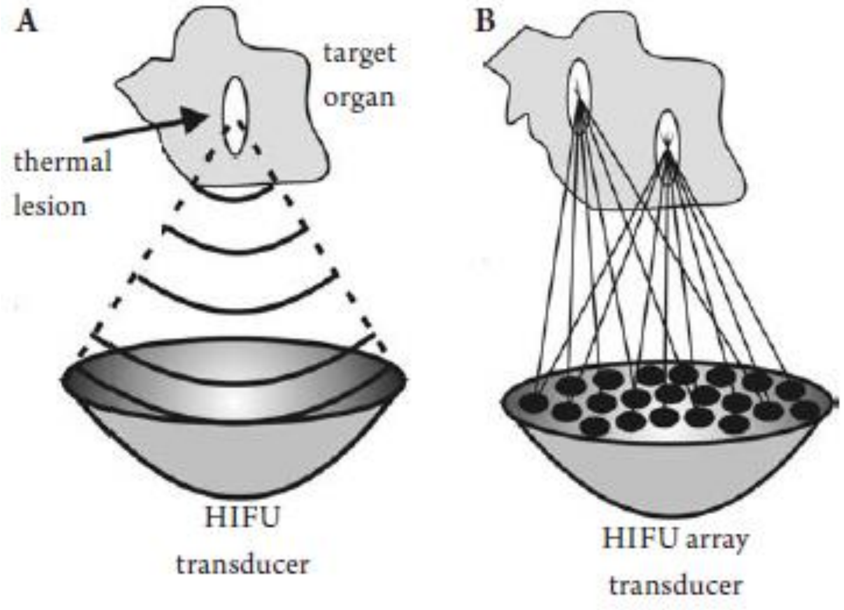
BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

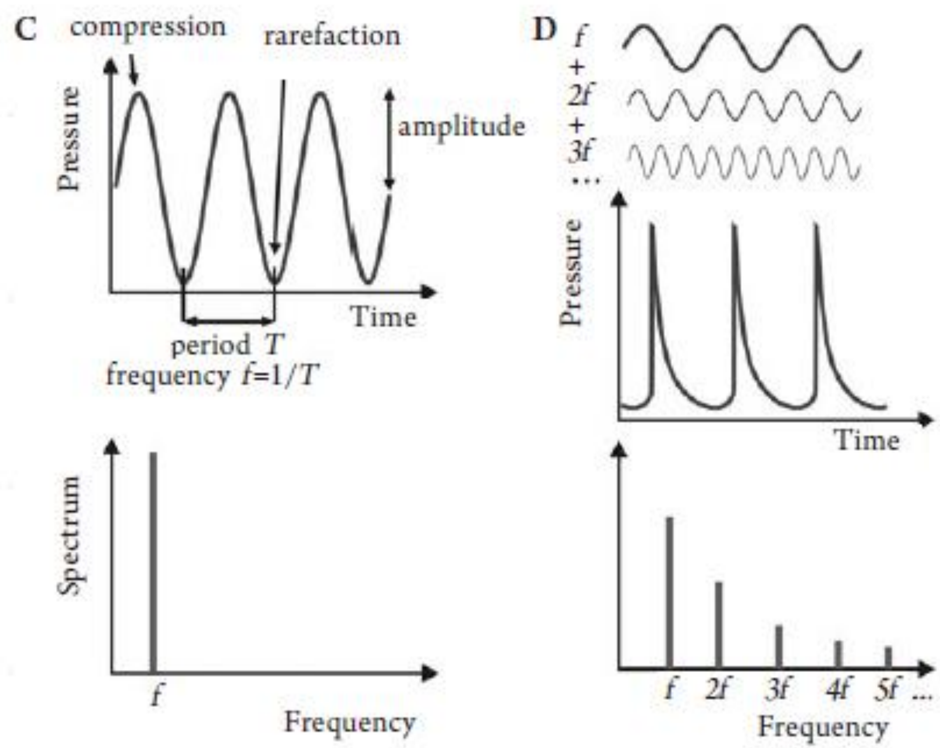
A way how to cook an egg in cold water



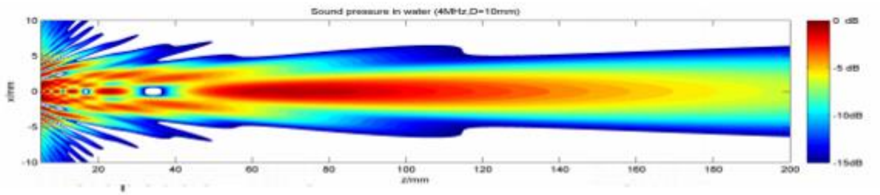


Thermal ablation, the most clinically advanced bioeffect of focused ultrasound, produces cell death in a targeted area with minimal damage to the surrounding tissue.

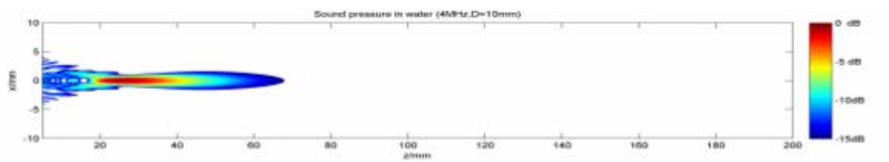


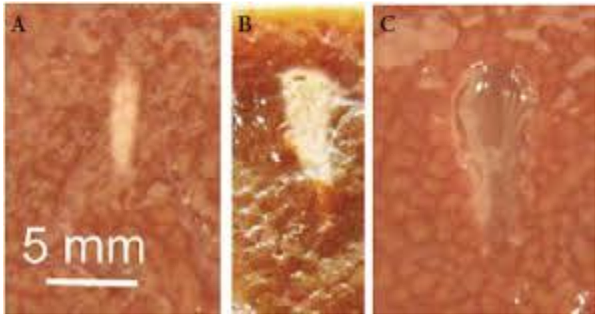


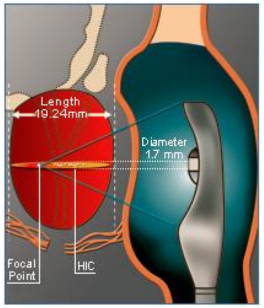
Diagnostic 1 – 20MHz

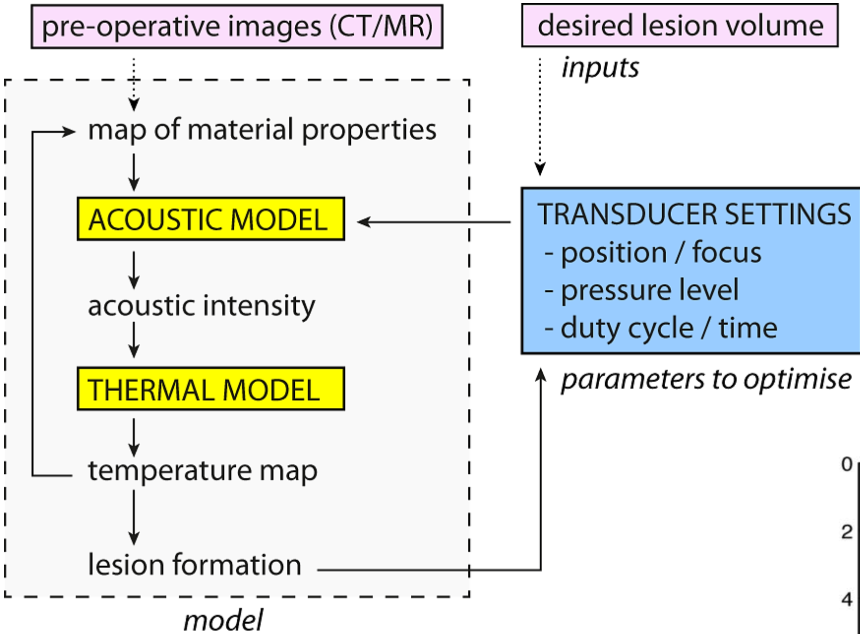


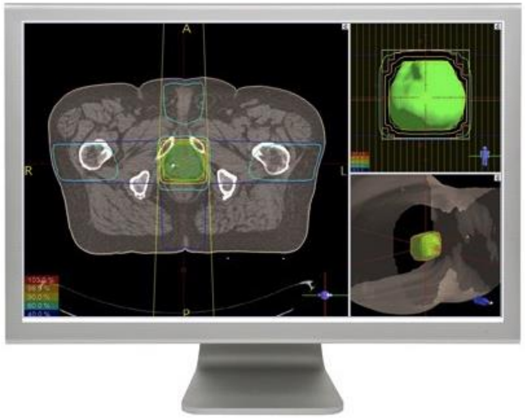
Ablative (HIFU) 1 – 5MHz

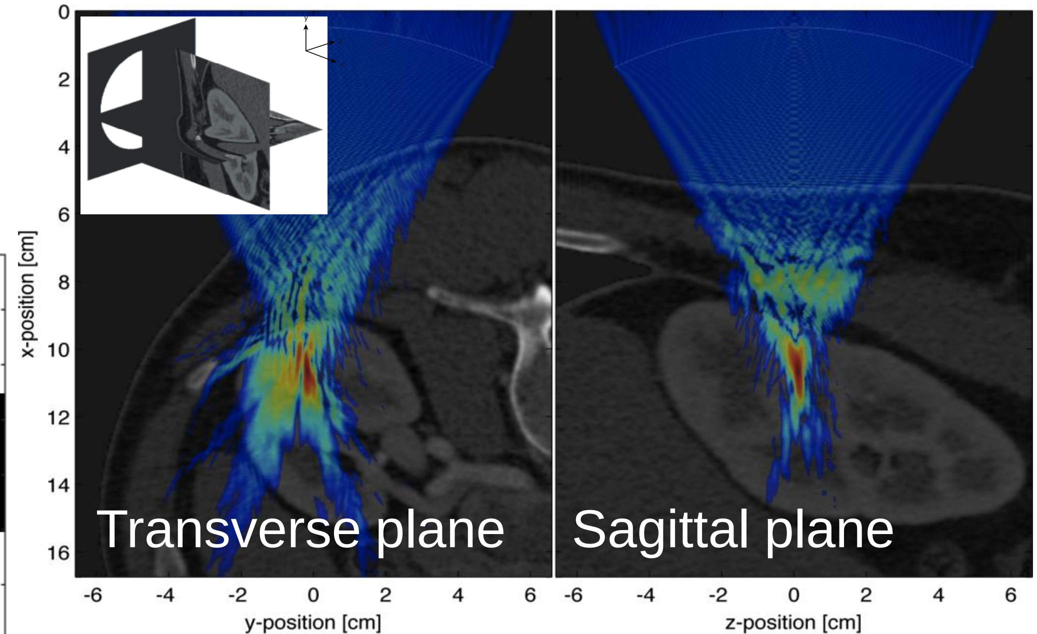


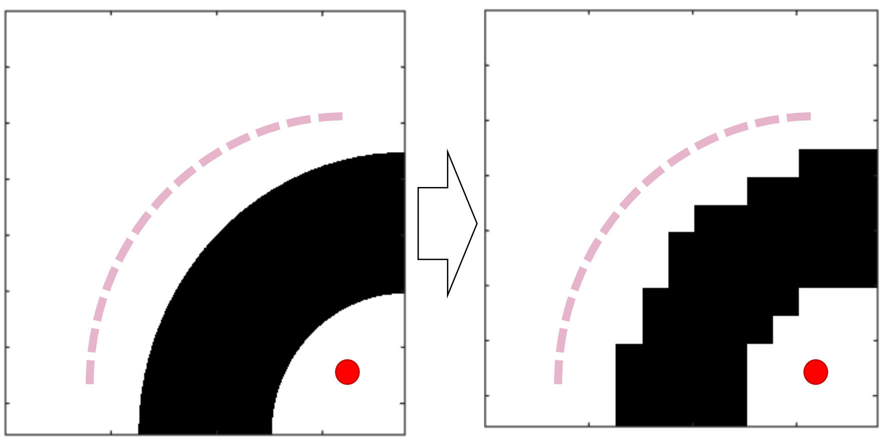





pre-operative images (CT/MR)
desired lesion volume
inputs
map of material properties
ACOUSTIC MODEL
acoustic intensity
THERMAL MODEL
temperature map
lesion formation
model
TRANSDUCER SETTINGS
- position / focus
- pressure level
- duty cycle / time
parameters to optimise




Transverse plane
Sagittal plane
x-position [cm]
y-position [cm]
z-position [cm]



- **k-Wave Toolbox** (http://www.k-wave.org)
  - 8000+ registered users
  - 3 base developers, 3 post-docs, 30 students
- **Full-wave 3D acoustic model**
  - including nonlinearity
  - heterogeneities
  - power law absorption
- **Solves coupled first-order equations**

$$\left\{\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} &= -\frac{1}{\rho_0}\nabla p \\ \frac{\partial \rho}{\partial t} &= -\left(2\rho + \rho_0\right)\nabla \cdot \mathbf{u} \\ p &= c_0^2\left(\rho + \frac{B}{2A}\frac{\rho^2}{\rho_0} - \mathrm{L}\rho\right) \end{aligned}\right.$$

momentum conservation

mass conservation

pressure-density relation

$$\mathrm{L} = \tau\frac{\partial}{\partial t}\left(-\nabla^2\right)^{\frac{y}{2}-1} + \eta\left(-\nabla^2\right)^{\frac{y+1}{2}-1}$$

absorption term

$$\frac{\partial}{\partial \xi} u_{\xi}^{n+1} = \mathbb{F}^{-1}\left\{ \kappa i k_{\xi} \mathbb{F}\left\{ u_{\xi}^{n} \right\} \right\}$$

- **Operations executed in every time step**
  - 6 forward 3D FFTs
  - 8 inverse 3D FFTs
  - *3+3 forward and inverse 1D FFTs in the case of non-staggered velocity*
  - About 100 element wise matrix operations (multiplication, addition,…)

- **Global data set**
  - 14 +3 (scratch) + *3 (unstaggering)* real 3D matrices
  - 3+3 complex 3D matrices
  - 6 real 1D vectors
  - 6 complex 1D vectors
  - *Sensor mask, source mask, source input*
  - *<0 , 20> real buffers for aggregated quantities (max, min, rms, max_all, min_all)*

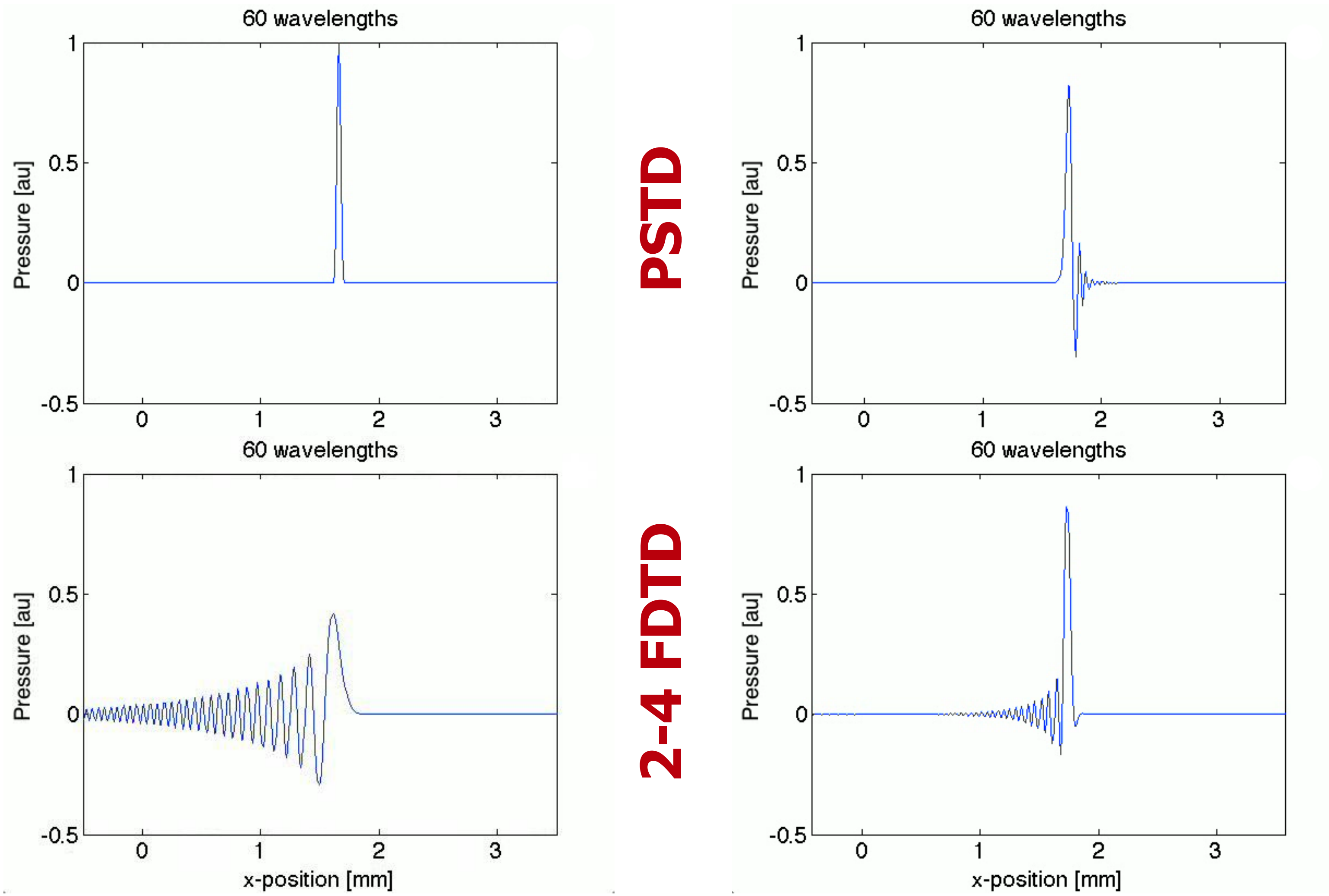
Pulse of $\sin^3(2\pi f_s t)$ propagation with 10 PPW

k-space PSTD

PSTD

2-2 FDTD

2-4 FDTD



Kidney and liver simulations (32 to 64 nodes)

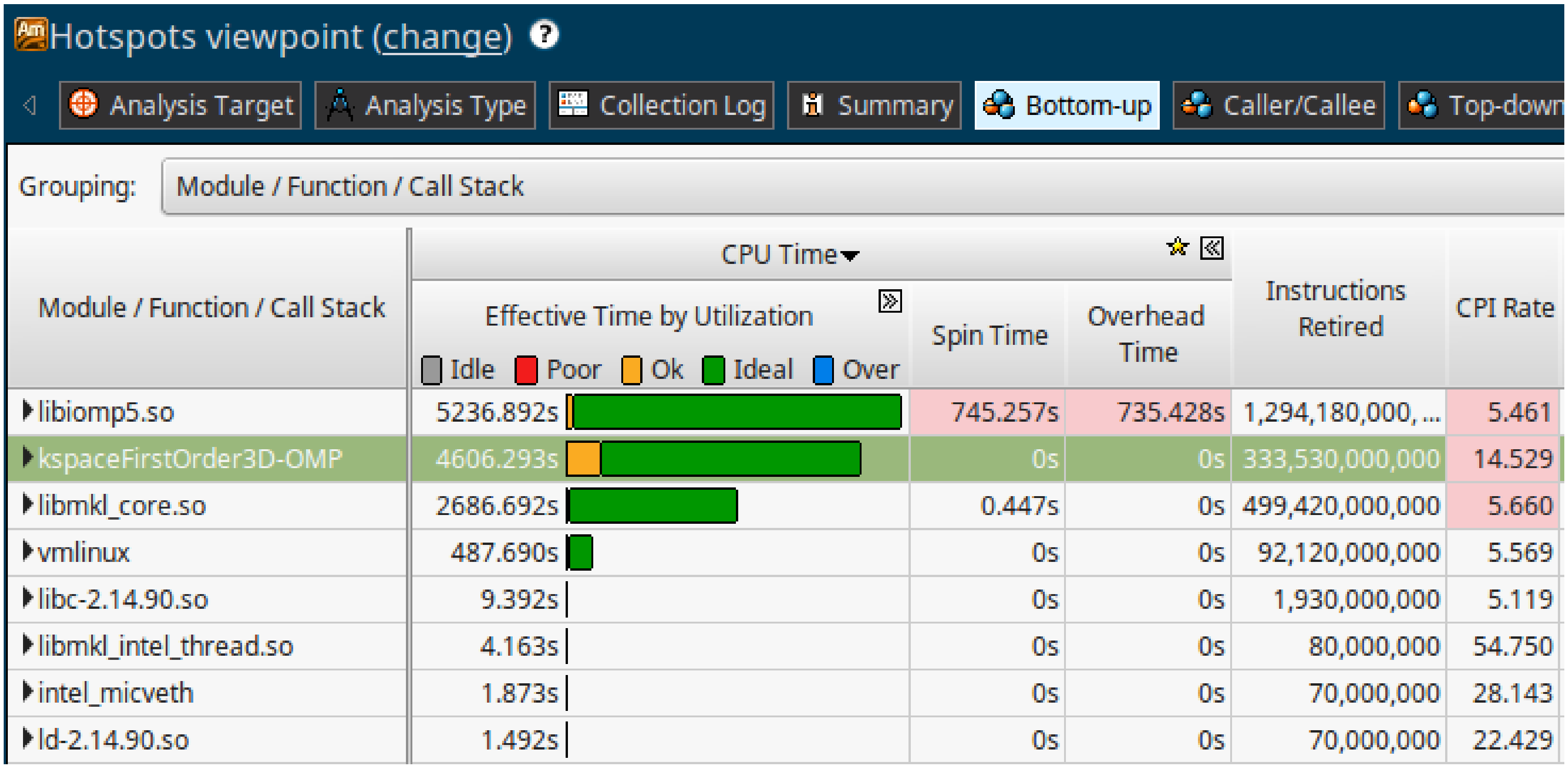
Medium	Grid Size	Properties	x m	t ns	fmax MHz	Time Steps	Mem MB	Compute Cores	Time d:hh:mm
Kideny-ref	1152 × 1152 × 1536	homogeneous	143	27.7	5.32	10,134	1.4	1024	0:09:14
Kidney	1152 × 1152 × 1536	heterogeneous	143	9.43	5.32	30,604	4.1	1536	0:20:42
Liver-ref	1344 × 1344 × 1792	homogeneous	123	23.8	6.21	11,823	2.3	768	0:18:10
Liver	1344 × 1344 × 1792	heterogeneous	123	8.06	6.21	35,801	6.6	1536	2:01:33
Water-ref	3072 × 3072 × 4096	homogeneous	53.7	10.3	14.2	27,139	3.1	1024	8:21:32

Prostate simulations (6 or 9 nodes, 150 simulations)

	Homogeneous Simulations				Heterogeneous Simulations			
Grid-size (pt^3)	RAM[a] (GB)	Input[b] (MB)	Output[b] (GB)	Time[c] (dd:hh:mm)	RAM[a] (GB)	Input[b] (GB)	Output[b] (GB)	Time[c] (dd:hh:mm)
$S_1 = 384 \times 256 \times 512$	10.5	2.9	0.7	00:00:10	11.9	4.1	2.0	00:00:39
$S_2 = 768 \times 512 \times 1024$	37.2	20.3	8.5	00:02:06	48.5	28.3	30.3	00:10:16
$S_3 = 1152 \times 768 \times 1536$	108.4	70.0	37.1	00:12:01	141.7	95.8	144.9	02:13:23
$S_4 = 1536 \times 1024 \times 2048$	246.4	165.2	108.4	01:16:07	331.9	225.9	445.9	08:11:38
$S_5 = 2304 \times 1536 \times 3072$	830.7	554.6	508.7	06:19:51	—	—	—	—

Profiling the Native Application on Xeon Phi

- Domain size: 256 x 256 x 256 ~ 2GB of RAM
- Original version, written in SSE 4.1 intrinsics



Hotspots viewpoint (change)

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down

Grouping: Module / Function / Call Stack

Module / Function / Call Stack	CPU Time: Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over)	CPU Time: Spin Time	CPU Time: Overhead Time	Instructions Retired	CPI Rate
libiomp5.so	5236.892s	745.257s	735.428s	1,294,180,000, ...	5.461
kspaceFirstOrder3D-OMP	4606.293s	0s	0s	333,530,000,000	14.529
libmkl_core.so	2686.692s	0.447s	0s	499,420,000,000	5.660
vmlinux	487.690s	0s	0s	92,120,000,000	5.569
libc-2.14.90.so	9.392s	0s	0s	1,930,000,000	5.119
libmkl_intel_thread.so	4.163s	0s	0s	80,000,000	54.750
intel_micveth	1.873s	0s	0s	70,000,000	28.143
ld-2.14.90.so	1.492s	0s	0s	70,000,000	22.429

Code Refactorisation for Xeon Phi

```
#pragma omp parallel
{
  // compute MAIN loop
  #pragma vector aligned
  #pragma omp simd
  #pragma omp for schedule (static) nowait
  for (size_t i = 0; i < TotalElementCount1; i++)
  {
    Register const float rho_xyz_el = rhox_data[i] + rhoy_data[i] + rhoz_data[i];
    RHO_Temp_Data[i]   = rho_xyz_el;
    BonA_Temp_Data[i]  = ((BonA[i * BonA_shift] * (rho_xyz_el * rho_xyz_el))     \
                         / (2.0f * rho0_data[i * rho0_shift])) + rho_xyz_el;
    SumDU_Temp_Data[i] = rho0_data[i * rho0_shift]                               \
                         * (dux_data[i] + duy_data[i] + duz_data[i]);
  }
```

```
#pragma vector nontemporal
#pragma omp simd aligned (rhox_data:64, rhoy_data:64, rhoz_data:64, rho0_data:64,    \
                          BonA:64, dux_data:64, duy_data:64, duz_data:64,            \
                          RHO_Temp_Data:64, BonA_Temp_Data:64, SumDU_Temp_Data:64)
#pragma omp for schedule (static)
for (size_t i = 0; i < RHO_Temp.GetTotalElementCount(); i++)
{
  register const float rho_xyz_el = rhox_data[i] + rhoy_data[i] + rhoz_data[i];
  RHO_Temp_Data[i]   = rho_xyz_el;
  BonA_Temp_Data[i]  = ((BonA[i * BonA_shift] * (rho_xyz_el * rho_xyz_el))       \
                       / (2.0f * rho0_data[i * rho0_shift])) + rho_xyz_el;
  SumDU_Temp_Data[i] = rho0_data[i * rho0_shift]                                 \
                       * (dux_data[i] + duy_data[i] + duz_data[i]);
}
```

Profiling the Optimised Version on Xeon Phi

- Domain size: 256 x 256 x 256 ~ 2GB of RAM
- Optimised code, written in OpenMP 4.0 intrinsics

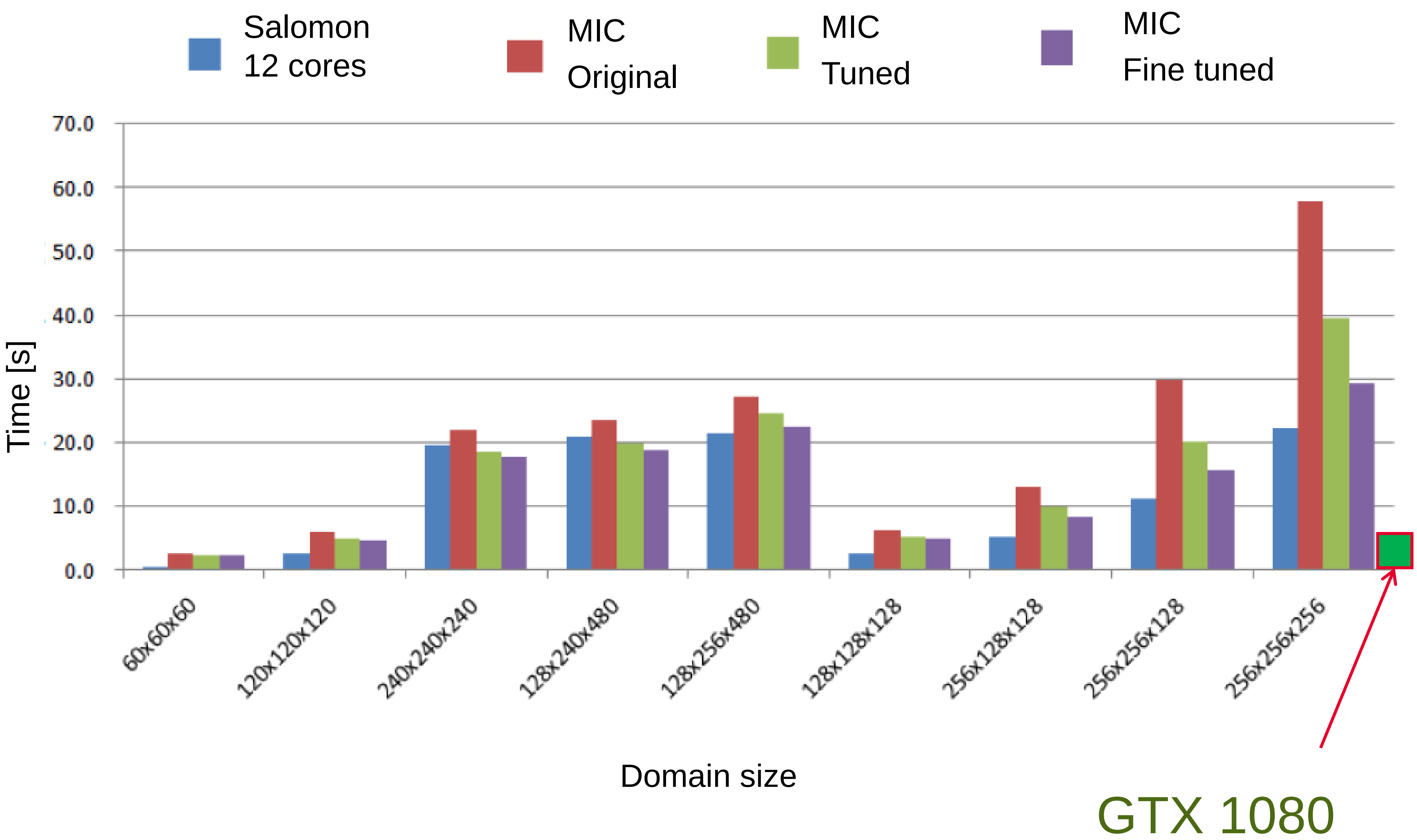
Hotspots viewpoint (change)

Analysis Target | Analysis Type | Collection Log | Summary | Bottom-up | Caller/Callee | Top-down Tree | Tasks and Frames

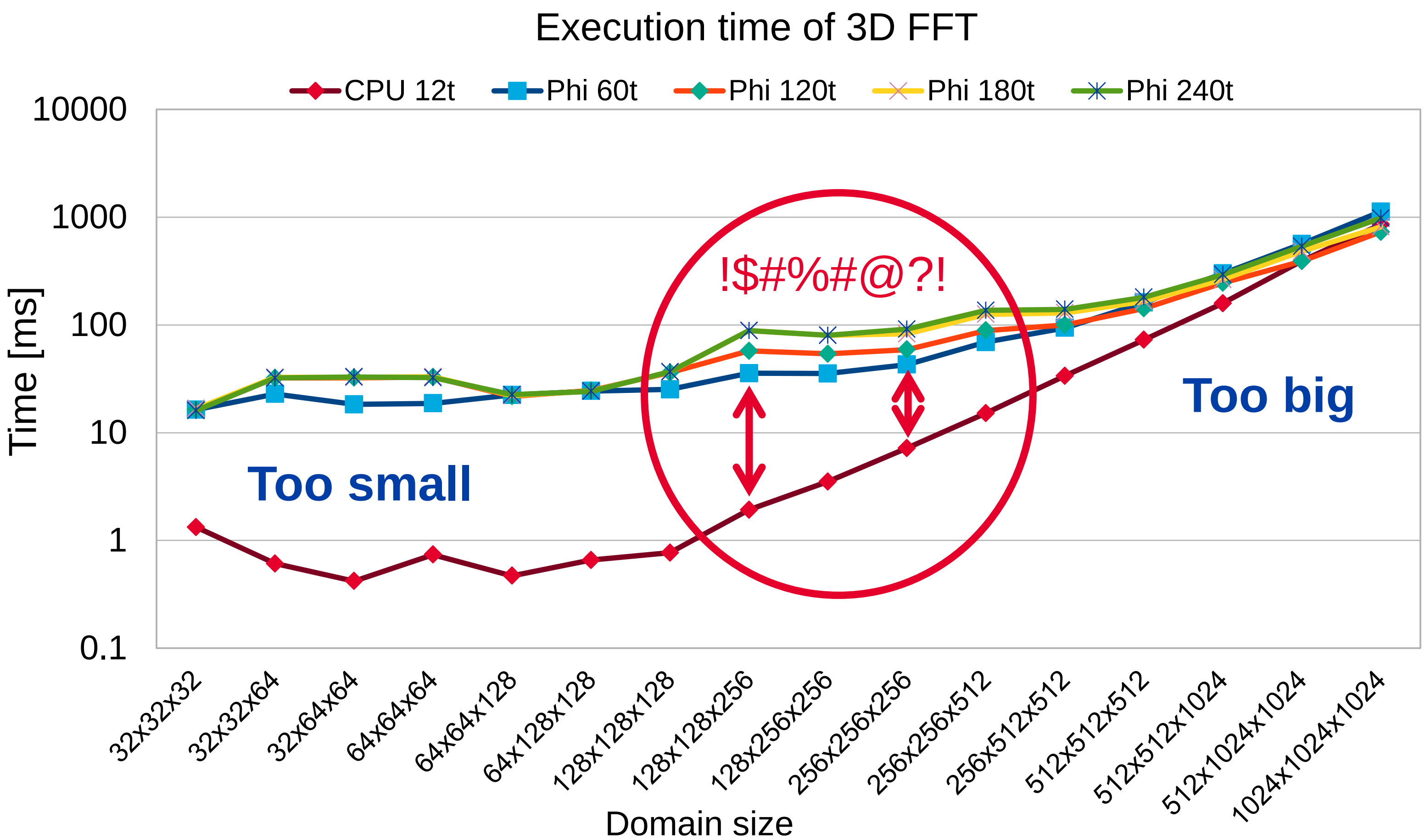
Grouping: Module / Function / Call Stack

Module / Function / Call Stack	CPU Time: Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over)	CPU Time: Spin Time	CPU Time: Overhead Time	Instructions Retired	CPI Rate
libiomp5.so	2941.702s	180.257s	174.211s	628,730,000,000	5.515
libmkl_core.so	2015.798s	0.342s	0s	488,820,000,000	4.339
kspaceFirstOrder3D-OMP	637.785s	0s	0s	110,610,000,000	6.066
TKSpaceFirstOrder3DSolver::Compute_rhoxyz_nonlinearompparallel@1287	109.791s	0s	0s	11,340,000,000	10.185
TKSpaceFirstOrder3DSolver::Compute_ddx_kappa_fft_pompparallel@1011	107.462s	0s	0s	28,510,000,000	3.965
TKSpaceFirstOrder3DSolver::Calculate_SumRho_BonA_SumDuompparallel@1650	93.964s	0s	0s	4,390,000,000	22.517
TKSpaceFirstOrder3DSolver::Compute_duxyzompparallel@1097	86.027s	0s	0s	31,720,000,000	2.853
TKSpaceFirstOrder3DSolver::Sum_Subterms_nonlinearompparallel@1919	70.827s	0s	0s	2,210,000,000	33.715
TKSpaceFirstOrder3DSolver::Compute_Absorb_nabla1_2	44.677s	0s	0s	17,540,000,000	2.680
Tuxyz_sgxyzMatrix::Compute_ux_sgx_normalize	39.192s	0s	0s	2,850,000,000	14.467
Tuxyz_sgxyzMatrix::Compute_uy_sgy_normalize	31.046s	0s	0s	2,110,000,000	15.479
Tuxyz_sgxyzMatrix::Compute_uz_sgz_normalize	28.650s	0s	0s	2,530,000,000	11.913

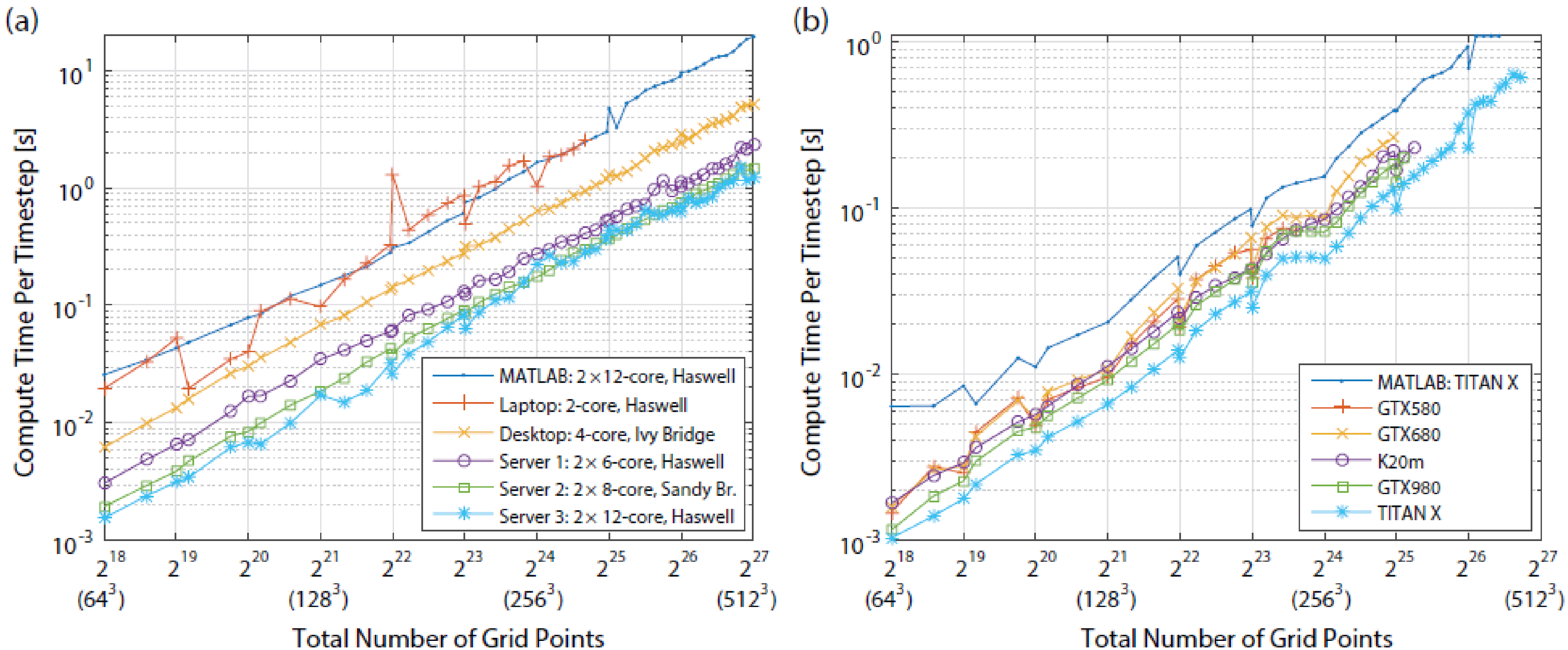
Performance Comparison



3D FFT Performance



Where We Are with GPU…



Maxwell TITAN-X 3.6x faster than a Salomon node

Native KNC on part with Matlab on Maxwell TITAN X

Conclusions

- KNC is not ready for k-Wave

- FFT-MKL performance is pretty poor for domain sizes of interest. Cache coherency problems?
- HDF5 I/O is terribly slow (forget about ZIP/SZIP compression)
- MPI implementation is always slower when KNC is enabled (both intranode and internode).

- Looking forward to KNL ☺

Comments and Questions

- The project is financed from the SoMoPro II programme. The research leading to this invention has acquired a financial grant from the People Programme (Marie Curie action) of the Seventh Framework Programme of EU according to the REA Grant Agreement No. 291782. The research is further co-financed by the South-Moravian Region. This work reflects only the author’s view and the European Union is not liable for any use that may be made of the information contained therein.
- This work was also supported by the research project "Architecture of parallel and embedded computer systems", Brno University of Technology, FIT-S-14-2297, 2014-2016.
- This work was supported by the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), funded by the European Regional Development Fund and the national budget of the Czech Republic via the Research and Development for Innovations Operational Programme, as well as Czech Ministry of Education, Youth and Sports via the project Large Research, Development and Innovations Infrastructures (LM2011033).
- We acknowledge CINECA and PRACE Summer of HPC project for the availability of high performance computing resources.
- The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (www.gauss-centre.eu) for funding this project by providing computing time on the GCS Supercomputer SuperMUC at Leibniz Supercomputing Centre (LRZ, www.lrz.de).

Thank you for your attention!

jarosjir@fit.vutbr.cz