THE UNIVERSITY OF TOKYO

東京大学情報基盤センター
INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

JCAHPC

# ℋACApK for Many-core Processors

Tetsuya Hoshino

## Overview of ℋACApK

- Fortran90 based library for H-matrices
- H-matrices: Approximation technique for dense matrices
- Provides parallel generation of H-matrix and parallel linear solvers

HACApK components

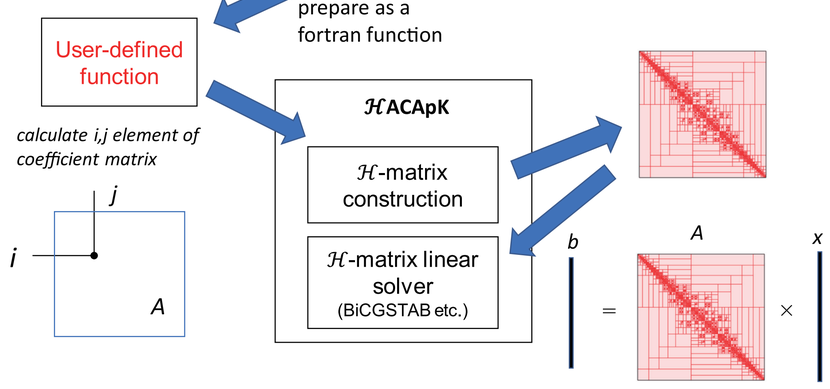
Target equation of BEM analysis:

$$\int_\Omega g(x,y)u(y)\mathrm{d}y = f \quad \text{discretization} \quad A\phi = b$$

$$A_{ij} = \int_\Omega \varphi_i(x)\int_\Omega g(x,y)\varphi(y)\mathrm{d}y\mathrm{d}x$$

$$b_i = \int_\Omega \varphi_i(x) f \mathrm{d}x$$

prepare as a fortran function

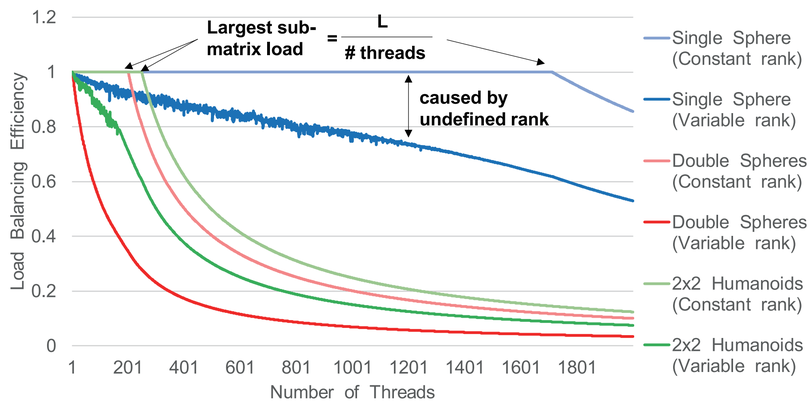



### Problems for many-core processors

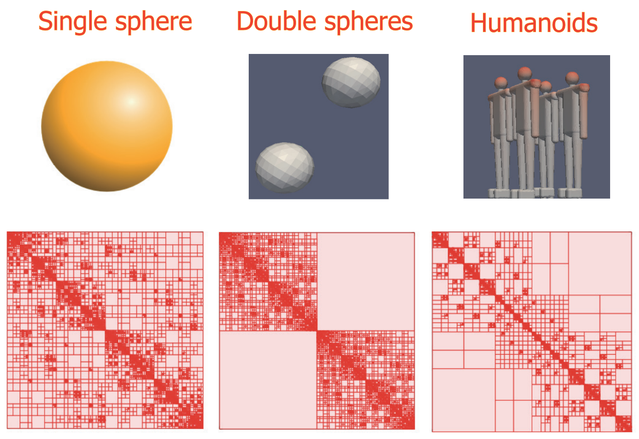
- Load balancing
  - The adaptive cross approximation (ACA) process is independently applied to all sub-matrices in original. However, the computational load of each sub-matrix is different and undefined before ACA process.
- SIMD vectorization
  - To use SIMD units, we have to vectorize black box user-defined function required for H-matrix construction.

Load balancing efficiency

$$E := \frac{(\textit{Total computational load})}{\#\ \text{threads} * (\textit{Max load of threads})}$$




Input data set of BEM problems and constructed H-matrices




## Load-balancing-aware Algorithms for GPUs

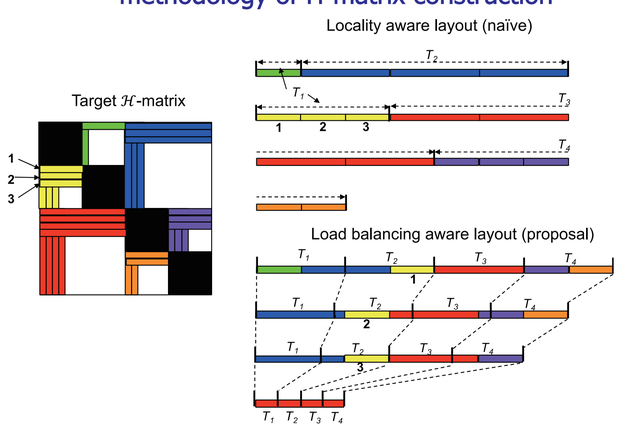
Tetsuya Hoshino, Akihiro Ida, Toshihiro Hanawa and Kengo Nakajima, " Load-balancing-aware Parallel Algorithms of H-matrices with Adaptive Cross Approximation for GPUs " , IEEE Cluster 2018, Belfast, UK (September 10-13, 2018)

- We propose load-balancing-aware H-matrix construction with ACA
  - ACA process is concurrently applied to all sub-matrices
- We implemented the proposed algorithm into HACApK
  - Evaluated on NVIDIA Tesla P100 and Intel Xeon Broadwell (BDW) with electrostatic field analyses

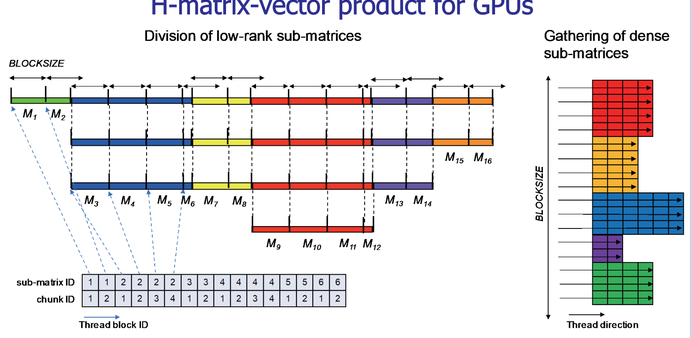
H-matrix construction with ACA

```
FOR ALL sub-matrices DO                 <- Loop interchange
choose first column vector
DO k = 1, KMAX
    FOA ALL column vector element DO
        calculate i, j element with userfunc
        subtract the summation of 1,2,...,k-1 element
    ENDDO
    calculate max of kth column
    FOA ALL row vector element DO
        calculate i, j element with userfunc
        subtract the summation of 1,2,...,k-1 element
    ENDDO
    calculate error
    IF(error < eps) EXIT
    calculate max of kth row
ENDDO
ENDDO
```

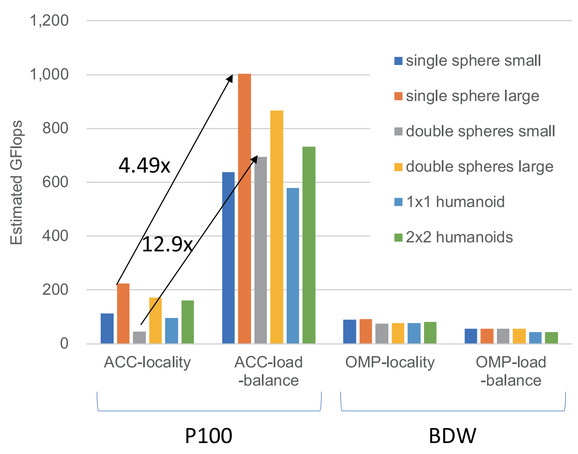
Memory ordering and parallelization methodology of H-matrix construction



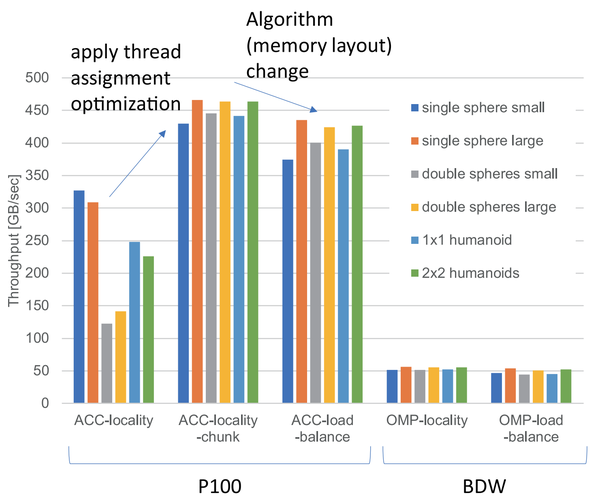

Parallelization methodology of H-matrix-vector product for GPUs




H-matrix construction




H-matrix vector product




## Semi-auto-vectorization for Wide SIMD Processors

Tetsuya Hoshino, Akihiro Ida, Toshihiro Hanawa and Kengo Nakajima, " Design of Parallel BEM Analyses Framework for SIMD Processors " , The International Conference on Computational Science 2018 (ICCS 2018), Wuxi, China

- We propose new design of user function interface of HACApK
  - Enhance compiler vectorization without SIMD knowledge
- We implemented the proposed design into HACApK
  - Evaluated on Intel Xeon Phi KNL and Intel Xeon Broadwell (BDW) with electrostatic field analyses

Parallelization of H-matrix construction with OpenMP directives

```
subrouine hmat_generation(st_bemv)
  type(hacapkInput) :: st_bemv
  !$omp parallel do
  do submat = 1, numSubmat
    ...
    !$omp simd                    <- is this SIMD directive work well?
    do i = str, enr
      a(i,j) = user_func(i,j,st_bemv)
    end do
    ...
  end do
end subroutine hmat_generation
```

User function caller

User function callee

**Original interface**

```
real(8) function user_func(i,j,st_bemv)
  integer :: i,j
  type(BemInput) :: st_bemv
  real(8) :: a1, a2, …

  a1 = st_bemv%a1(i,j)
  a2 = st_bemv%a2(i,j)
  …
  ! calculate i,j value of coefficient

end function user_func
```

User implementation

**Data access**

```
subroutine set_args(i,j,st_bemv, a1, a2, …)
  integer :: i,j
  type(hacapkInput) :: st_bemv
  real(8) :: a1, a2, …
  a1 = st_bemv%a1(i,j)
  a2 = st_bemv%a2(i,j)
  …
end subroutine set_args
```

Structure including input model data

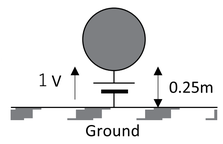
**Computation**

```
real(8) function vectorize_func(a1, a2, …)
!$omp declare simd(vectorize_func)          &
!$omp simdlen(SIMDLENGTH)                   &
!$omp linear(ref(a1, a2, …))
  real(8) :: a1, a2, …
  ! calculate i,j value of coefficient
end function vectorize_func
```

Automatically inserted by framework

```
real(8),dimension(SIMDLENGTH) :: ans,arg1,arg2,...
...
  do i = str,enr,SIMDLENGTH
    do ii = 1,SIMDLENGTH
      call set_args(i,j,st_bemv,arg1(ii),arg2(ii),...)
    end do
    !$omp simd
    do ii = 1,SIMDLENGTH
      ans(ii) = vectorize_func(arg1(ii),arg2(ii),...)
    end do
    do ii = 1,SIMDLENGTH
      a(i,j) = ans(ii)
    end do
  end do
...
```

Electrostatic field analysis with single sphere conductor

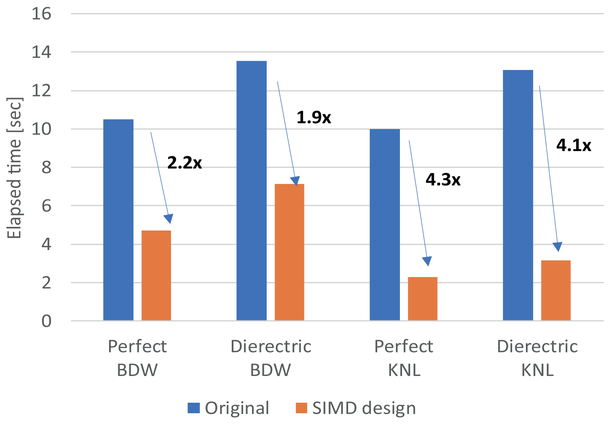



- We use two user-defined functions
  - Pefect conductor P[u] without branch
  - Dierectric D[u] with branch divergent

$$\mathcal{P}[u](x) := \int_\Omega \frac{1}{4\pi||x-y||}u(y)\mathrm{d}y, x \in \Omega$$

$$\mathcal{D}[u](x) := \int_\Omega \frac{\langle x-y, n(y)\rangle}{4\pi||x-y||^3}u(y)\mathrm{d}y, x \in \Omega$$

Coefficient H-matrix generation on BDW and KNL




Contact: Hoshino Tetsuya <hoshino@cc.u-tokyo.ac.jp>
http://www.cspp.cc.u-tokyo.ac.jp/hoshino/