UPC++ for Bioinformatics: A Case Study Using Genome-Wide Association Studies

Jan C. Kässens*,**Jorge González-Domínguez****,
Lars Wienbrandt*,Bertil Schmidt**

*Department of Computer Science, Christian-Albrechts-University of Kiel, Germany
{jka,lwi}@informatik.uni-kiel.de
**Parallel and Distributed Architectures Group, Johannes Gutenberg University of Mainz, Germany
{j.gonzalez,bertil.schmidt}@uni-mainz.de

IEEE International Conference on Cluster Computing
Cluster 2014

JG|U

C|A|U

Genome-Wide Association Studies (I)

Analyses of genetic influence
on diseases

Genome-Wide Association Studies (I)

Analyses of genetic influence on diseases

- *M* individuals

Genome-Wide Association Studies (I)

Analyses of genetic influence on diseases

- *M* individuals
  - *K* cases

Genome-Wide Association Studies (I)

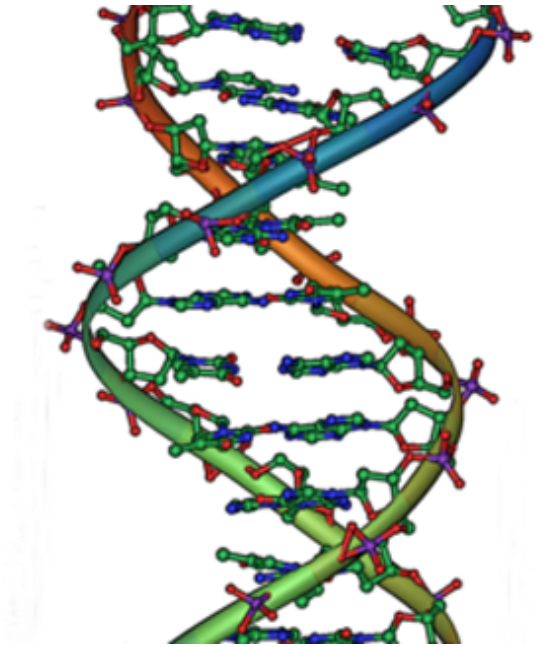
Analyses of genetic influence on diseases

- M individuals
  - K cases
  - C controls

Genome-Wide Association Studies (I)

Analyses of genetic influence on diseases

- *M* individuals
  - *K* cases
  - *C* controls
- *N* genetic markers, Single Nucleotide Polymorphisms (SNPs). 3 genotypes:
  - Homozygous Wild (w, AA, 0)
  - Heterozygous (h, Aa, 1)
  - Homozygous Variant (v, aa, 2)



Genome-Wide Association Studies (II)

	Cases							Controls							
SNP 1	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	1
SNP 2	0	1	1	0	2	0	0	0	1	2	2	1	0	1	1	2
SNP 3	0	0	0	0	0	0	0	0	1	2	1	1	1	2	1	1
SNP 4	0	1	0	1	0	1	0	1	2	2	2	2	1	1	1	1
SNP 5	0	2	2	2	0	1	1	1	1	0	0	1	1	0	2	2
SNP 6	1	0	1	0	1	0	1	0	1	2	1	2	1	2	2	1

Genome-Wide Association Studies (II)

	Cases						Controls									
SNP 1	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	1
SNP 2	0	1	1	0	2	0	0	0	1	2	2	1	0	1	1	2
SNP 3	0	0	0	0	0	0	0	0	1	2	1	1	1	2	1	1
SNP 4	0	1	0	1	0	1	0	1	2	2	2	2	1	1	1	1
SNP 5	0	2	2	2	0	1	1	1	1	0	0	1	1	0	2	2
SNP 6	1	0	1	0	1	0	1	0	1	2	1	2	1	2	2	1

Genome-Wide Association Studies (II)

	Cases						Controls									
SNP 1	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	1
SNP 2	0	1	1	0	2	0	0	0	1	2	2	1	0	1	1	2
SNP 3	0	0	0	0	0	0	0	0	1	2	1	1	1	2	1	1
SNP 4	0	1	0	1	0	1	0	1	2	2	2	2	1	1	1	1
SNP 5	0	2	2	2	0	1	1	1	1	0	0	1	1	0	2	2
SNP 6	1	0	1	0	1	0	1	0	1	2	1	2	1	2	2	1

Genome-Wide Association Studies (and III)

Definition

Two SNPs present epistasis or interaction if:

- Their joint genotype frequencies show a statistically significant difference between cases and controls which potentially explains the effect of the genetic variation leading to disease.
- The difference between cases and controls shown by the joint values is significantly higher than using only the individual SNP values.

BOOST

BOolean Operation-based Screening and Testing

- Binary traits
- Exhaustive search
- Statistical regression
- Good accuracy (used by biologists)
- Returns a list of SNP pairs with high interaction probability
- Fastest available tool. Intel Core i7 3.20GHz:
  - 40,000 SNPs and 3,200 individuals
    - About 800 million pairs
    - 51 minutes
  - 500,000 SNPs and 5,000 individuals
    - About 125 billion pairs (moderated size)
    - Estimated 7 days

GBOOST

CUDA version for GPUs

- Same accuracy as BOOST
- 40,000 SNPs and 6,400 individuals
  - About 800 million pairs
  - 28 seconds on a GTX Titan
- 500,000 SNPs and 5,000 individuals
  - About 125 billion pairs (moderated size)
  - 1 hour on a GTX Titan

GBOOST

CUDA version for GPUs

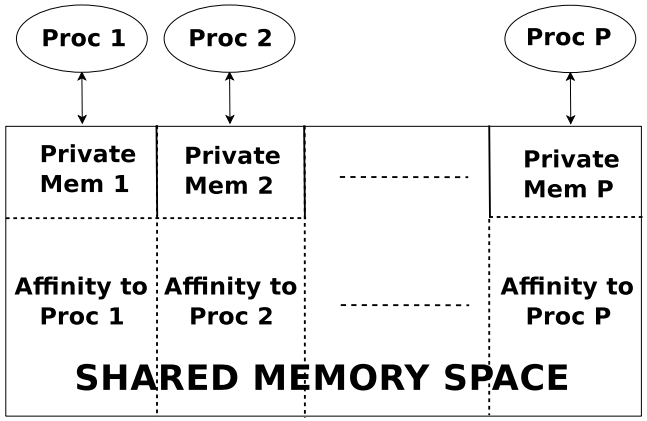
- Same accuracy as BOOST
- 40,000 SNPs and 6,400 individuals
  - About 800 million pairs
  - 28 seconds on a GTX Titan
- 500,000 SNPs and 5,000 individuals
  - About 125 billion pairs (moderated size)
  - 1 hour on a GTX Titan

High-throughput genotyping technologies collect few million SNPs of an individual within a few minutes $\rightarrow$ Expected datasets with 5M SNPs and 10,000 individuals

UPC++ (I)

- Unified Parallel C++
- Novel extension of ANSI C++
  - Y Zheng, A Kamil, M Driscoll, H Shan, and K Yelick. `UPC++: a PGAS Extension for C++`. *In Proc. 28th IEEE Intl. Parallel and Distributed Processing Symp. (IPDPS'14)*, Phoenix, AR, USA, 2014.
- Follows the Partitioned Global Address Space (PGAS) programming model
- Single Program Multiple Data (SPMD) execution model
- Works on shared and distributed memory systems

UPC++ (and II)



- Global memory logically partitioned among processes
- Processes can directly access (read/write) any part of the global memory
- Memory with affinity usually mapped in the same node (faster accesses)

Creation of Contingency Tables (I)

For each SNP-pair $\rightarrow$ Number of occurrences of each combination of genotypes

Cases	**SNP2=0**	**SNP2=1**	**SNP2=2**
SNP1=0	n_{000}	n_{010}	n_{020}
SNP1=1	n_{100}	n_{110}	n_{120}
SNP1=2	n_{200}	n_{210}	n_{220}
Controls	**SNP2=0**	**SNP2=1**	**SNP2=2**
SNP1=0	n_{001}	n_{011}	n_{021}
SNP1=1	n_{101}	n_{111}	n_{121}
SNP1=2	n_{201}	n_{211}	n_{221}

Creation of Contingency Tables (and II)

SNP 4	0	1	0	1	0	1	0	1	2	2	2	2	1	1	1	1
SNP 6	1	0	1	0	1	0	1	0	1	2	1	2	1	2	2	1

Cases	SNP6=0	SNP6=1	SNP6=2
SNP4=0	0	4	0
SNP4=1	4	0	0
SNP4=2	0	0	0
Controls	**SNP6=0**	**SNP6=1**	**SNP6=2**
SNP4=0	0	0	0
SNP4=1	0	2	2
SNP4=2	0	1	2

Filtering Stage (I)

Measuring interaction via log-linear models

Filtering Stage (I)

Measuring interaction via log-linear models

Log-Linear Measure (I)

$$\hat{L}_S - \hat{L}_H = N \sum_{ijk} \left[\hat{\pi}_{ijk} \log \left(\frac{\hat{\pi}_{ijk}}{\hat{p}_{ijk}} \right) \right]$$

- $\hat{L}_S$ log-likelihood of the saturated regression model
- $\hat{L}_H$ log-likelihood of the homogeneous association model
- $\hat{\pi}_{ijk}$ joint distribution obtained under the saturated model
- $\hat{p}_{ijk}$ distribution obtained under the homogeneous association model

Filtering Stage (II)

Measuring interaction via log-linear models

Log-Linear Measure (II)

$$\hat{L}_S - \hat{L}_H = N \sum_{ijk} \left[\hat{\pi}_{ijk} \log \left(\frac{\hat{\pi}_{ijk}}{\hat{p}_{ijk}} \right) \right]$$

- T the threshold for epistasis
- If $\hat{L}_S - \hat{L}_H > T \Rightarrow$ Epistasis

Filtering Stage (II)

Measuring interaction via log-linear models

Log-Linear Measure (II)

$$\hat{L}_S - \hat{L}_H = N \sum_{ijk} \left[\hat{\pi}_{ijk} \log \left(\frac{\hat{\pi}_{ijk}}{\hat{p}_{ijk}} \right) \right]$$

- T the threshold for epistasis
- If $\hat{L}_S - \hat{L}_H > T \Rightarrow$ Epistasis
- Computationally expensive
  - $\hat{p}_{ijk}$ computed through iterative methods

Filtering Stage (III)

Kirkwood Superposition Approximation (KSA)

- $\hat{L}_S - \hat{L}_{KSA} = N \sum_{ijk} \left[\hat{\pi}_{ijk} \log \left(\frac{\hat{\pi}_{ijk}}{\hat{p}^k_{ijk}} \right) \right]$
- $\hat{p}^k_{ijk} = \frac{1}{\eta} \frac{\pi_{ij.} \pi_{i.k} \pi_{.jk}}{\pi_{i..} \pi_{.j.} \pi_{..k}}$
- $\eta = \sum_{ijk} \frac{\pi_{ij.} \pi_{i.k} \pi_{.jk}}{\pi_{i..} \pi_{.j.} \pi_{..k}}$

Filtering Stage (III)

Kirkwood Superposition Approximation (KSA)

- $\hat{L}_S - \hat{L}_{KSA} = N \sum_{ijk} \left[\hat{\pi}_{ijk} \log \left(\frac{\hat{\pi}_{ijk}}{\hat{p}^k_{ijk}} \right) \right]$
- $\hat{p}^k_{ijk} = \frac{1}{\eta} \frac{\pi_{ij.} \pi_{i.k} \pi_{.jk}}{\pi_{i..} \pi_{.j.} \pi_{..k}}$
- $\eta = \sum_{ijk} \frac{\pi_{ij.} \pi_{i.k} \pi_{.jk}}{\pi_{i..} \pi_{.j.} \pi_{..k}}$
- Upper bound: $\hat{L}_S - \hat{L}_H \leq \hat{L}_S - \hat{L}_{KSA}$

Filtering Stage (III)

Kirkwood Superposition Approximation (KSA)

- $\hat{L}_S - \hat{L}_{KSA} = N \sum_{ijk} \left[\hat{\pi}_{ijk} \log \left(\frac{\hat{\pi}_{ijk}}{\hat{p}^k_{ijk}} \right) \right]$
- $\hat{p}^k_{ijk} = \frac{1}{\eta} \frac{\pi_{ij.} \pi_{i.k} \pi_{.jk}}{\pi_{i..} \pi_{.j.} \pi_{..k}}$
- $\eta = \sum_{ijk} \frac{\pi_{ij.} \pi_{i.k} \pi_{.jk}}{\pi_{i..} \pi_{.j.} \pi_{..k}}$
- Upper bound: $\hat{L}_S - \hat{L}_H \leq \hat{L}_S - \hat{L}_{KSA}$
- $\hat{L}_S - \hat{L}_{KSA} < T \Rightarrow$ No epistasis

Filtering Stage (III)

Kirkwood Superposition Approximation (KSA)

- $\hat{L}_S - \hat{L}_{KSA} = N \sum_{ijk} \left[\hat{\pi}_{ijk} \log \left(\frac{\hat{\pi}_{ijk}}{\hat{p}^k_{ijk}} \right) \right]$
- $\hat{p}^k_{ijk} = \frac{1}{\eta} \frac{\pi_{ij.} \pi_{i.k} \pi_{.jk}}{\pi_{i..} \pi_{.j.} \pi_{..k}}$
- $\eta = \sum_{ijk} \frac{\pi_{ij.} \pi_{i.k} \pi_{.jk}}{\pi_{i..} \pi_{.j.} \pi_{..k}}$
- Upper bound: $\hat{L}_S - \hat{L}_H \leq \hat{L}_S - \hat{L}_{KSA}$
- $\hat{L}_S - \hat{L}_{KSA} < T \Rightarrow$ No epistasis
- $\hat{L}_S - \hat{L}_{KSA}$ is computationally simpler and faster

Filtering Stage (and IV)

Pseudocode

Filtering Stage (and IV)

Pseudocode

- For each SNP-pair *P*
  1. Calculate Contingency Table of P

Filtering Stage (and IV)

Pseudocode

- For each SNP-pair P
  1. Calculate Contingency Table of P
  2. $v = KSA\_Value(P)$

Filtering Stage (and IV)

Pseudocode

- For each SNP-pair P
  1. Calculate Contingency Table of P
  2. $v = KSA\_Value(P)$
  3. If $v > T$
     1. $v = LogLinear\_Value(P)$

Filtering Stage (and IV)

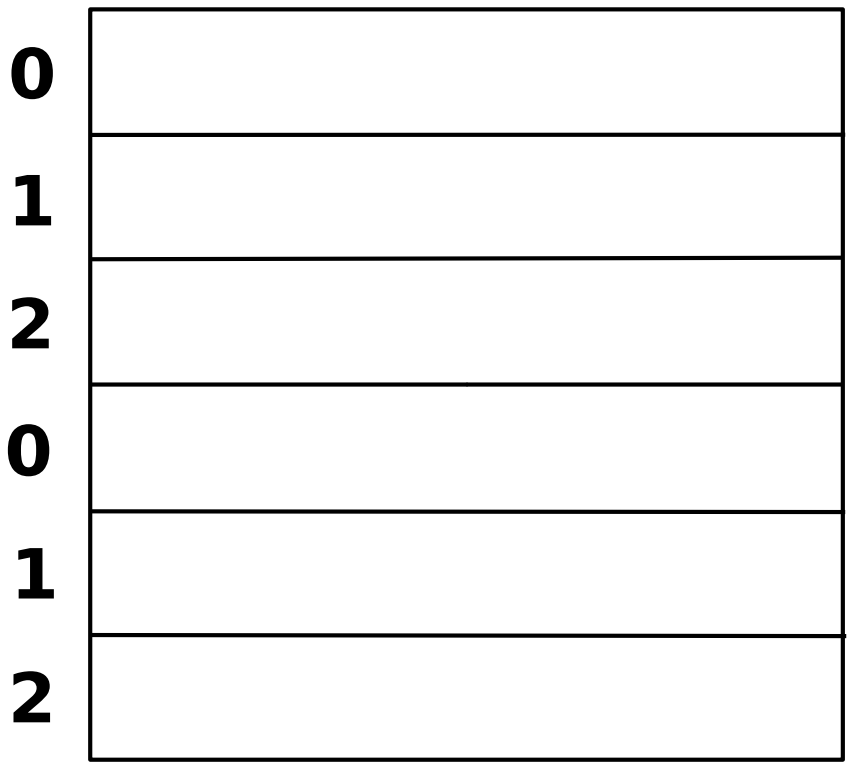
Pseudocode

- For each SNP-pair P
  1. Calculate Contingency Table of P
  2. $v = KSA\_Value(P)$
  3. If $v > T$
     1. $v = LogLinear\_Value(P)$
     2. If $v > T$ include P in the output list as pair with epistasis

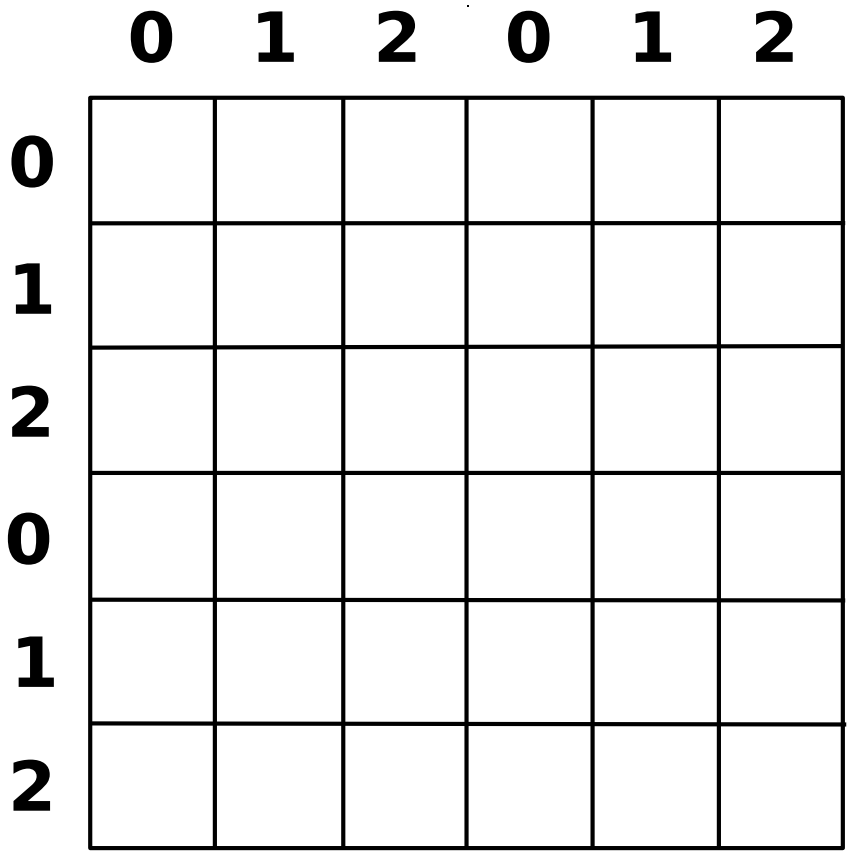
Data Distribution (I)

- Information of the N SNPs distributed in block-cyclic way
- $P \rightarrow$ number of UPC++ processes
- $numBP \rightarrow$ configurable number of blocks per process
- $NB = numBP * P \rightarrow$ total number of blocks
- $NMB = \frac{NB*(NB-1)}{2} \rightarrow$ number of metablocks (blocks of SNP-pairs)
  - Possible combinations of blocks of SNPs

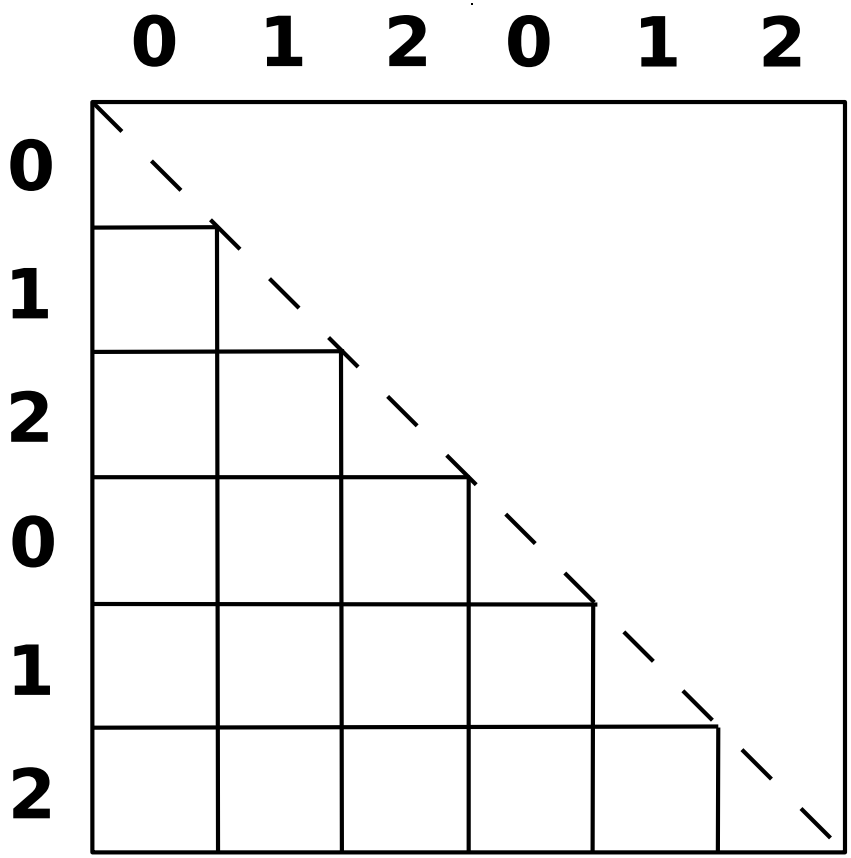
Data Distribution (II)



Data Distribution (II)



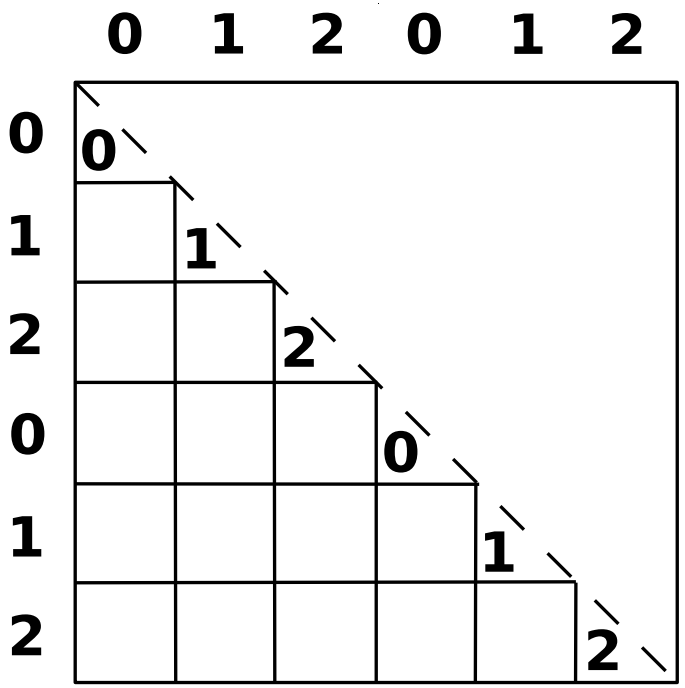
Data Distribution (II)



Data Distribution (III)

Assignment of Metablocks

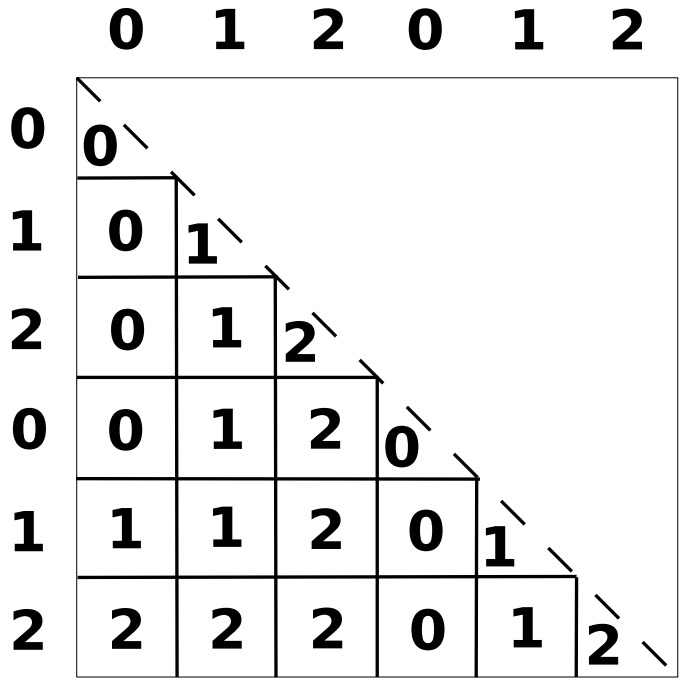
- Diagonal to processes with all information in local memory



Data Distribution (III)

Assignment of Metablocks

- Diagonal to processes with all information in local memory
- Rectangular to processes with at least one block in local memory (row or column of the matrix)



Data Distribution (and IV)

Goal of the Distribution

- Balance of the workload
  - Similar number of metablocks per UPC++ process
- Minimization of remote copies
  - At least one of the blocks of biallelic information already in local memory

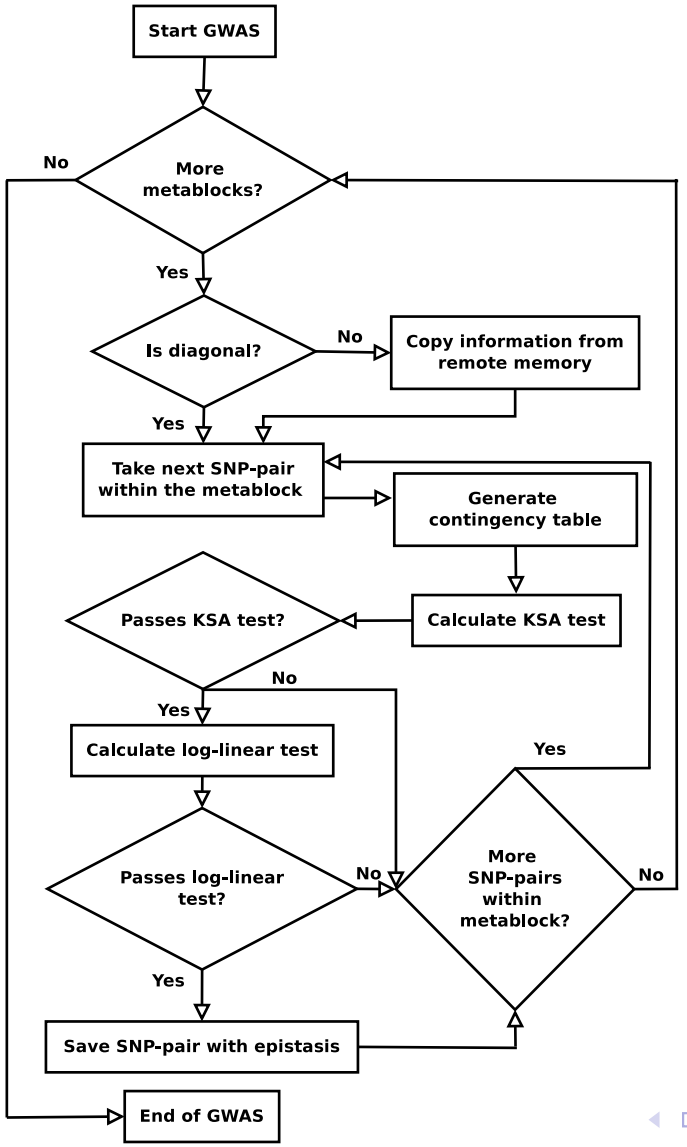
Data Distribution (and IV)

Goal of the Distribution

- Balance of the workload
  - Similar number of metablocks per UPC++ process
- Minimization of remote copies
  - At least one of the blocks of biallelic information already in local memory

- Other distributions that fulfill these conditions can be used

Process Workflow



UPC++ Optimization Techniques

- Data locality exploitation
  - Minimization of remote copies

UPC++ Optimization Techniques

- Data locality exploitation
  - Minimization of remote copies
- Overlapping computation and communication
  - Asynchronous communications

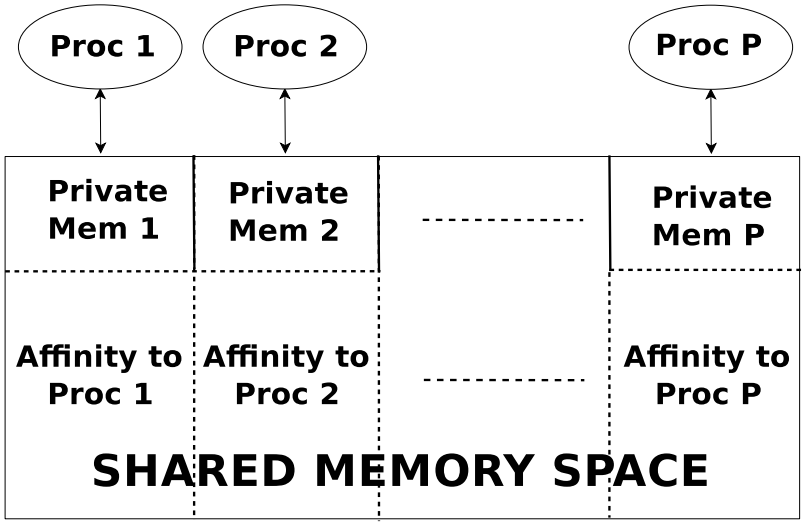
UPC++ Optimization Techniques

- Data locality exploitation
  - Minimization of remote copies
- Overlapping computation and communication
  - Asynchronous communications
- Aggregation of remote memory accesses
  - Bulk copies instead of data one-by-one

UPC++ Optimization Techniques

- Data locality exploitation
  - Minimization of remote copies
- Overlapping computation and communication
  - Asynchronous communications
- Aggregation of remote memory accesses
  - Bulk copies instead of data one-by-one
- Space privatization

UPC++ Optimization Techniques



UPC++ Optimization Techniques

- Data locality exploitation
  - Minimization of remote copies
- Overlapping computation and communication
  - Asynchronous communications
- Aggregation of remote memory accesses
  - Bulk copies instead of data one-by-one
- Space privatization
  - Usage of regular C++ pointers to access the local part of global memory
  - UPC++ global pointers save information:
    - Which part of the memory is pointed
    - Address within the block
  - Updating C++ pointers is faster

Hybrid UPC++&PThreads Implementation (I)

Options in Multicore Clusters

1. One UPC++ process per core
2. One UPC++ process per node
   - Exploit multicore with PThreads
3. Intermediate options

Hybrid UPC++&PThreads Implementation (I)

Options in Multicore Clusters

1. One UPC++ process per core
2. One UPC++ process per node
   - Exploit multicore with PThreads
3. Intermediate options

Hybrid Approach

- Variable number of UPC++ processes per node
- Distribute metablocks among UPC++ processes
- Each UPC++ process launches T PThreads
- Pairs within the metablock distributed among the PThreads

Hybrid UPC++&PThreads Implementation (II)

Advantage

- Less UPC++ processes
  - Less blocks and metablocks
  - Less remote data copies

Drawback

- Computing each metablock requires synchronizations among the PThreads

Hybrid UPC++&PThreads Implementation (and III)

Distributions among PThreads

1. Static distribution
   - SNP-pairs assigned to PThreads at the beginning of the metablock computation $\rightarrow$ block distribution
   - All PThreads compute the same number of pairs
   - Some PThreads might wait for other with more pairs that pass the KSA filter

Hybrid UPC++&PThreads Implementation (and III)

Distributions among PThreads

1. Static distribution
   - SNP-pairs assigned to PThreads at the beginning of the metablock computation $\rightarrow$ block distribution
   - All PThreads compute the same number of pairs
   - Some PThreads might wait for other with more pairs that pass the KSA filter
2. Dynamic distribution
   - The SNP-pairs are associated to SNPs on demand
   - When a PThread finishes the computation of a group of pairs looks for another group
   - PThreads are not idle
   - Mutex or semaphore needed to synchronize accesses to the variables that indicate which SNP-pairs have been analyzed

Platforms

Pluton

- University of A Coruña (UDC)
- 16 nodes
  - 2 Intel Xeon E5-2660 Sandy Bridge processors
    - 8 cores at 2.20 Ghz
- FDR InfiniBand network

Edison

- National Energy Research Supercomputing Center (NERSC)
- 18th in the June 2014 TOP500 list
- 5,576 nodes
  - 2 Intel Xeon E5-4603 Ivy Bridge
    - 12 cores at 2.40 Ghz
- Cray Aries network
  - Dragonfly topology

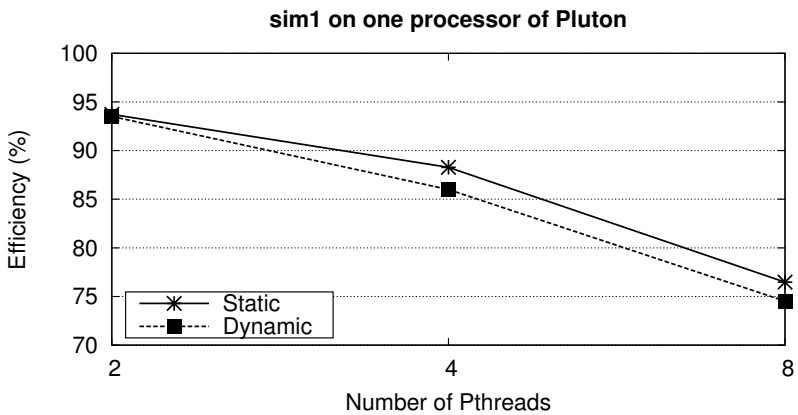
Datasets

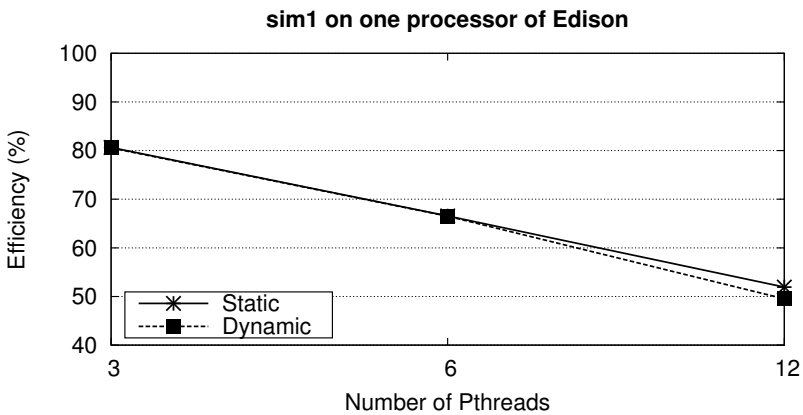
	sim1	**sim2**	**sim3**	**WTCCCIbd**
Real data?	No	No	No	Yes
Number of SNPs	10,000	25,000	50,000	500,568
Number of cases	800	1,600	1,600	2,005
Number of controls	800	1,600	1,600	3,004
Seq. time *Pluton*	2 m	19 m	1 h 15 m	$>$ 6 days
Seq. time *Edison*	1 m	10 m	41 m	$>$ 3 days

Best PThreads Distribution (I)

	sim1	**sim2**	**sim3**	**WTCCCIbd**
Real data?	**No**	No	No	Yes
Number of SNPs	**10,000**	25,000	50,000	500,568
Number of cases	**800**	1,600	1,600	2,005
Number of controls	**800**	1,600	1,600	3,004
Seq. time *Pluton*	**2 m**	19 m	1 h 15 m	$>$ 6 days
Seq. time *Edison*	**1 m**	10 m	41 m	$>$ 3 days

Best PThreads Distribution (and II)





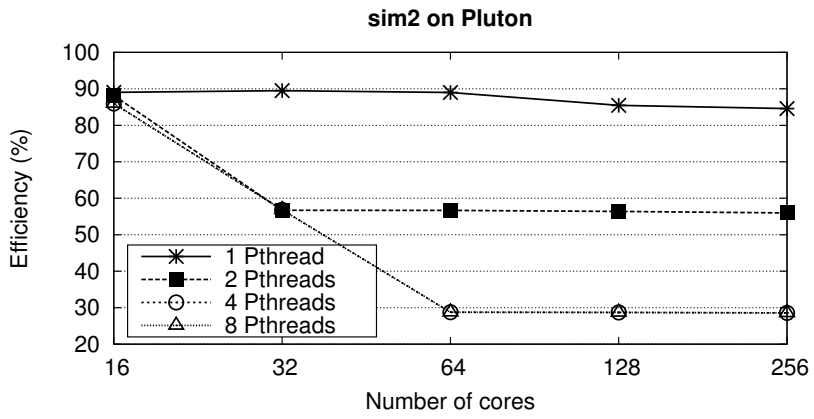
Best Number of PThreads (I)

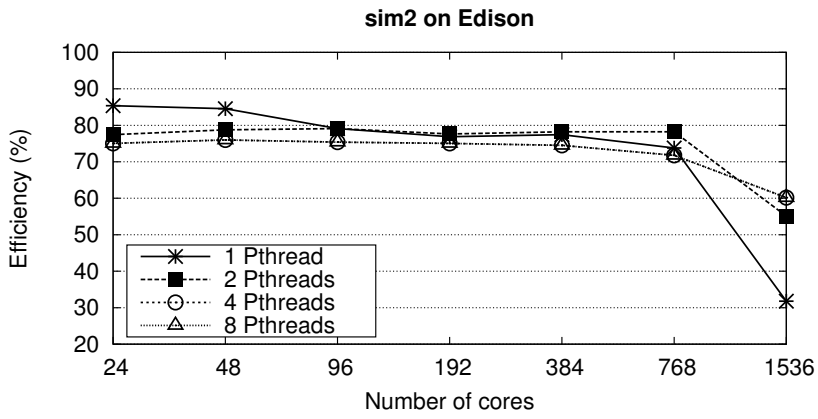
	sim1	**sim2**	**sim3**	**WTCCCIbd**
Real data?	No	**No**	**No**	Yes
Number of SNPs	10,000	**25,000**	**50,000**	500,568
Number of cases	800	**1,600**	**1,600**	2,005
Number of controls	800	**1,600**	**1,600**	3,004
Seq. time *Pluton*	2 m	**19 m**	**1 h 15 m**	$>$ 6 days
Seq. time *Edison*	1 m	**10 m**	**41 m**	$>$ 3 days

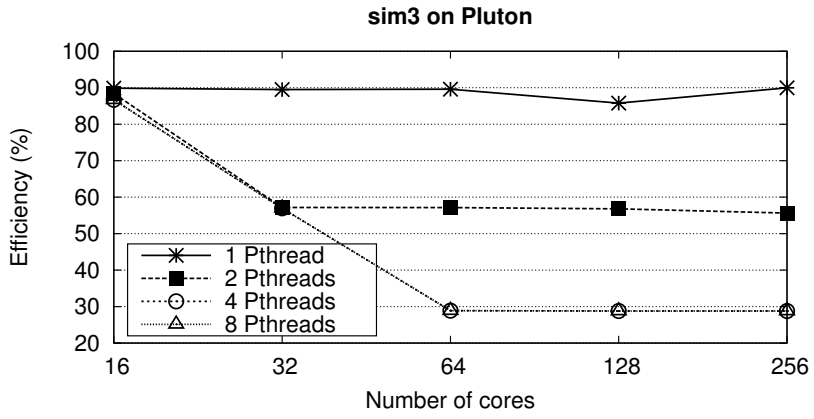
Configuration

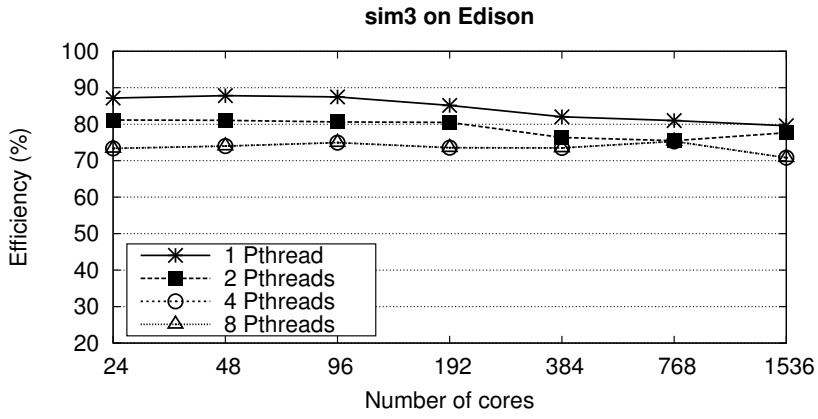
- Static distribution for PThreads
- The best number of blocks per process

Best Number of PThreads (and II)









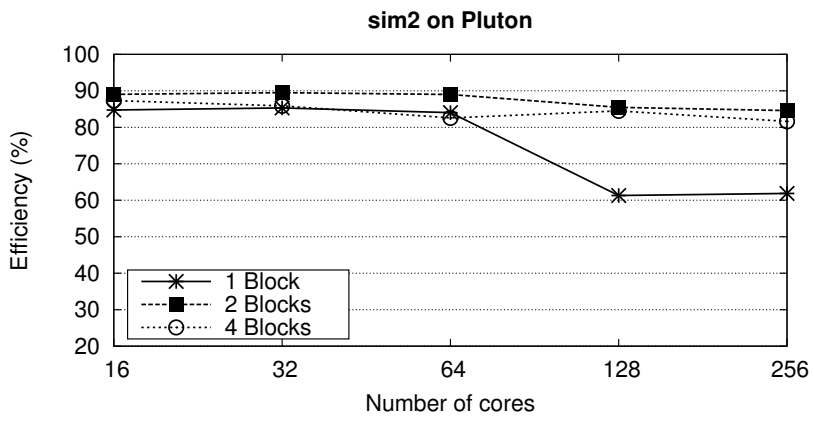
Best Number of Blocks per Process (I)

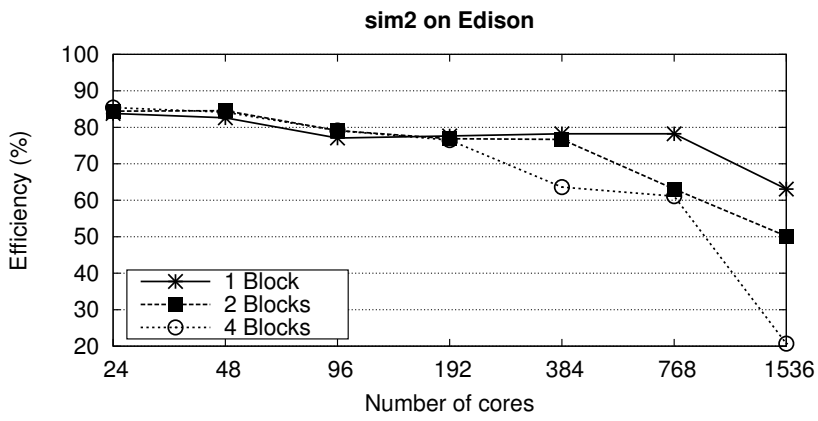
	sim1	**sim2**	**sim3**	**WTCCCIbd**
Real data?	No	**No**	**No**	Yes
Number of SNPs	10,000	**25,000**	**50,000**	500,568
Number of cases	800	**1,600**	**1,600**	2,005
Number of controls	800	**1,600**	**1,600**	3,004
Seq. time *Pluton*	2 m	**19 m**	**1 h 15 m**	$>$ 6 days
Seq. time *Edison*	1 m	**10 m**	**41 m**	$>$ 3 days

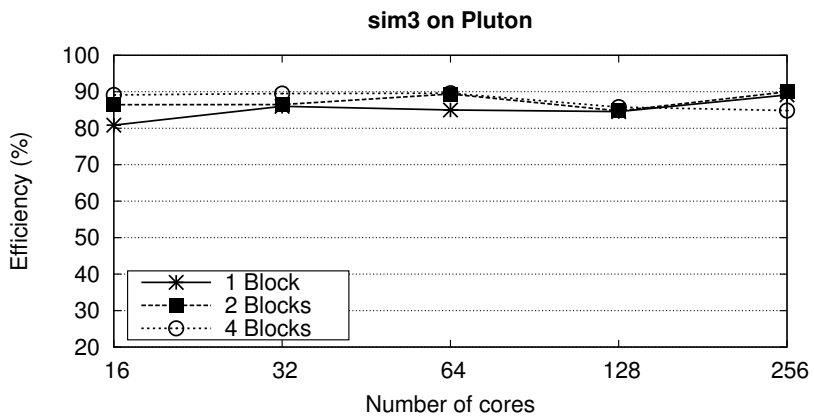
Configuration

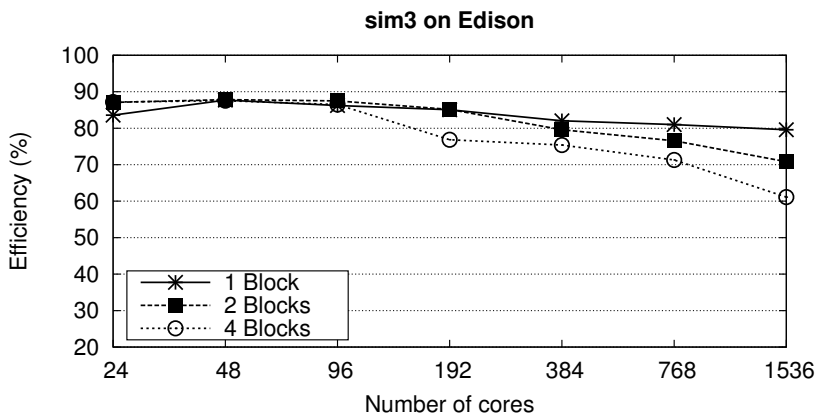
- Static distribution for PThreads
- The best number of UPC++ processes per node

Best Number of Blocks per Process (and II)









Evaluation of the Configurable Parameters

- Always static distribution for PThreads

Evaluation of the Configurable Parameters

- Always static distribution for PThreads
- Hybrid implementation useful for strong scaling
  - Not large datasets
  - Many cores
- One UPC++ process per core on other scenarios

Evaluation of the Configurable Parameters

- Always static distribution for PThreads
- Hybrid implementation useful for strong scaling
  - Not large datasets
  - Many cores
- One UPC++ process per core on other scenarios
- 1 block per process for large number of cores
  - Try to avoid generating more remote copies
  - Less significant influence on large datasets
- 2 or 4 blocks per process for moderate number of cores
  - Improvement is not very significant

Comparison with Other Parallel Architectures

	sim1	**sim2**	**sim3**	**WTCCCIbd**
Real data?	No	No	No	**Yes**
Number of SNPs	10,000	25,000	50,000	**500,568**
Number of cases	800	1,600	1,600	**2,005**
Number of controls	800	1,600	1,600	**3,004**
Seq. time *Pluton*	2 m	19 m	1 h 15 m	$>$ **6 days**
Seq. time *Edison*	1 m	10 m	41 m	$>$ **3 days**

Platform	**Time**
Edison (12,288 cores)	45 s
Edison (1,536 cores)	5 m
Pluton (256 cores)	45 m
NVIDIA GTX Titan	1h 01 m
NVIDIA GTX 750Ti	2h 41 m

Summary

- Tool to search for epistasis between SNP-pairs in a fast manner exploiting clusters and supercomputers
  - Based on regression model
- Developed using UPC++ (first bioinformatics implementation)
  - Inheritance and polymorphism from OOP
  - Takes advantage of one-sided communication from PGAS
  - Distribution and optimization techniques to minimize communication overhead
- Hybrid UPC++&PThreads implementation
- Same accuracy and faster than the popular (G)BOOST
  - GBOOST $\rightarrow$ 61 m for the WTCCC dataset on a GTX Titan
  - 45 m on the 256 cores of`Pluton` (1.35x)
  - 45 s on 12,288 cores of `Edison` (81.33x)
- Future work: UPC++&CUDA tool for multi-GPU

UPC++ for Bioinformatics: A Case Study Using Genome-Wide Association Studies

Jan C. Kässens*,**Jorge González-Domínguez****,
Lars Wienbrandt*,Bertil Schmidt**

*Department of Computer Science, Christian-Albrechts-University of Kiel, Germany
{jka,lwi}@informatik.uni-kiel.de
**Parallel and Distributed Architectures Group, Johannes Gutenberg University of Mainz, Germany
{j.gonzalez,bertil.schmidt}@uni-mainz.de

IEEE International Conference on Cluster Computing
Cluster 2014

JG|U

C|A|U