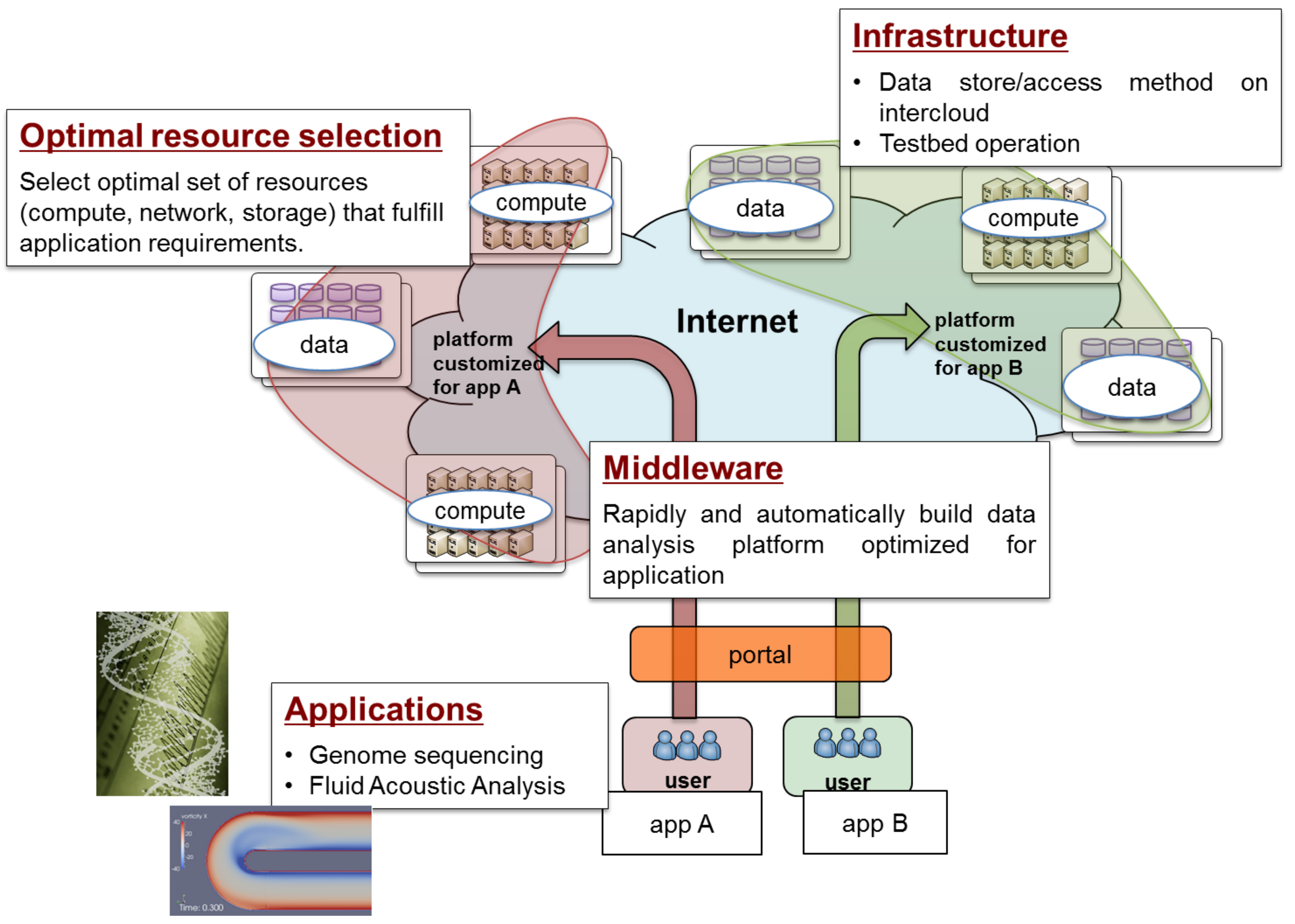
Optimal Resource Selection for Intercloud Workflows

INFORMATION INITIATIVE CENTER
Hokkaido University

CREST Application-Centric Cloud Overlay Project



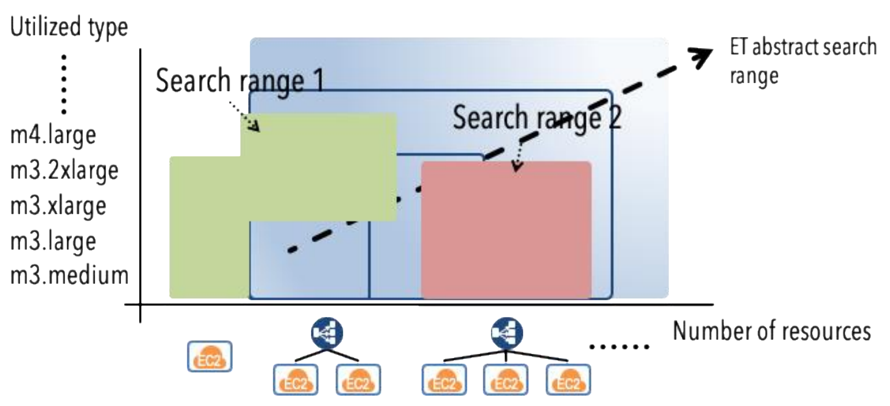
Deliverables for Optimal Resource Selection

1. **Multi-objective optimization** that can handle more than 10 (3-5 => 30-50) objective measures such as performance, cost, SLA, and greenness while also satisfying hundreds of requirements and constraints.
2. **Highly efficient formal method for descriptive specification** of virtualized systems and the intercloud environment.
3. Dynamic optimization/adaptation mechanism considering resilience, fault tolerance, changing environment, and user requirements.

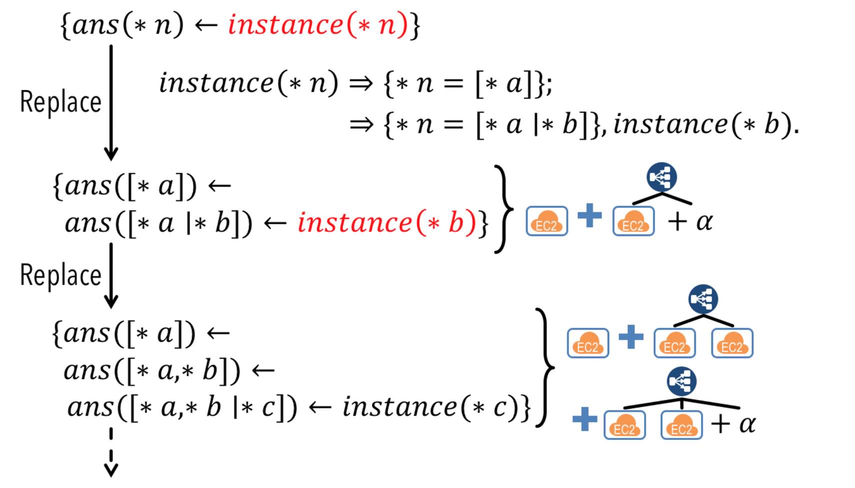
Equivalent Transformation (ET) Constraint Solver

The **Constraint Solver** is based on Equivalent transformation (ET).
ET Program: Set of declarative meaning-preserving rewriting rules.
ET Solver: Finds all feasible solutions to a problem based on a logic search tree.

- SAT-based methods: Search range must be specified (*e.g., Search range 1 or 2).*
- ET-based method: Search range can be abstractly determined such as "Combination of two or more instances".



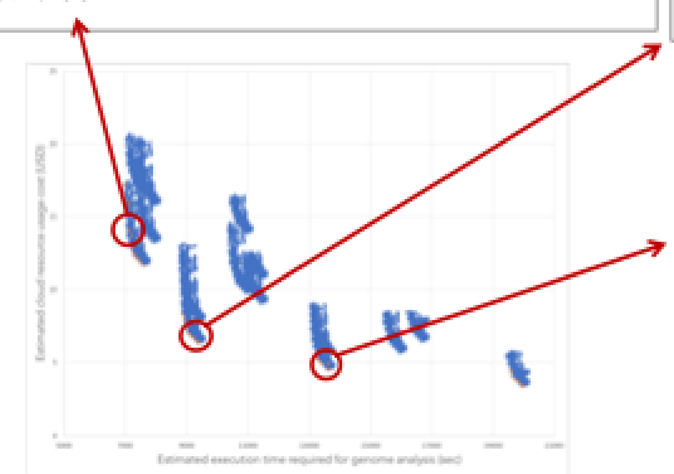
Advantage: Generate application infrastructures composed of any number of resources by branching clause state.



Bi-objective Optimization via MOPLS

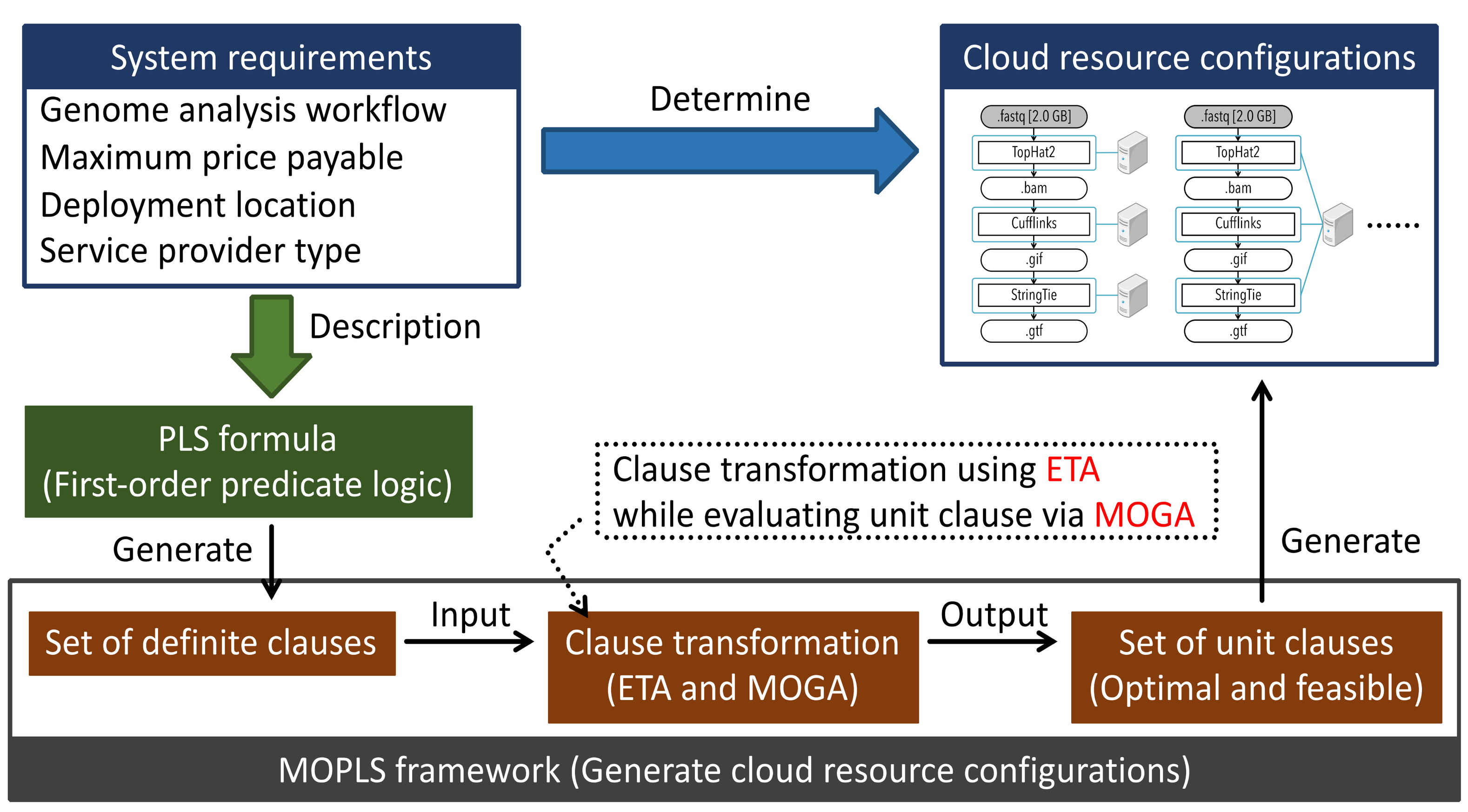
[Cloud resource configuration A]
Deploy **m4.16xlarge** in Virginia region for TopHat2
Deploy **c4.8xlarge** in Virginia region for Cufflinks
Deploy **c4.2xlarge** in Virginia region for StringTie
[Execution time for genome analysis]
7145.17 sec
[Cloud resource usage cost]
13.18 USD

[Cloud resource configuration B]
Deploy **c4.8xlarge** in Virginia region for TopHat2
Deploy **c4.2xlarge** in Virginia region for Cufflinks
Deploy **c4.xlarge** in Virginia region for StringTie
[Execution time for genome analysis]
9357.24 sec
[Cloud resource usage cost]
6.32 USD

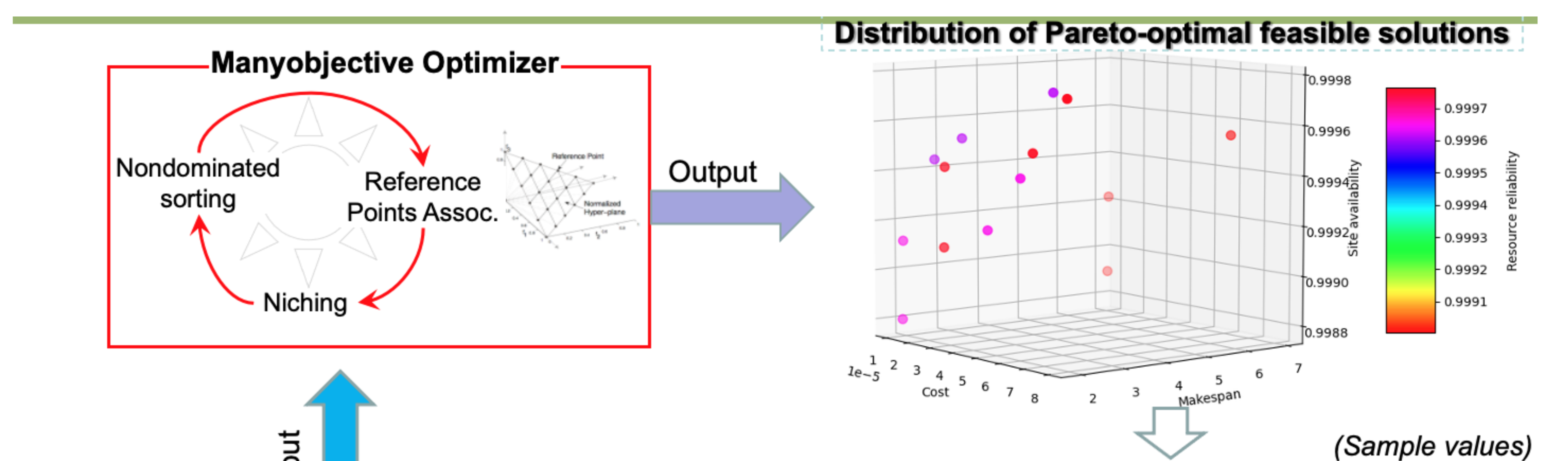


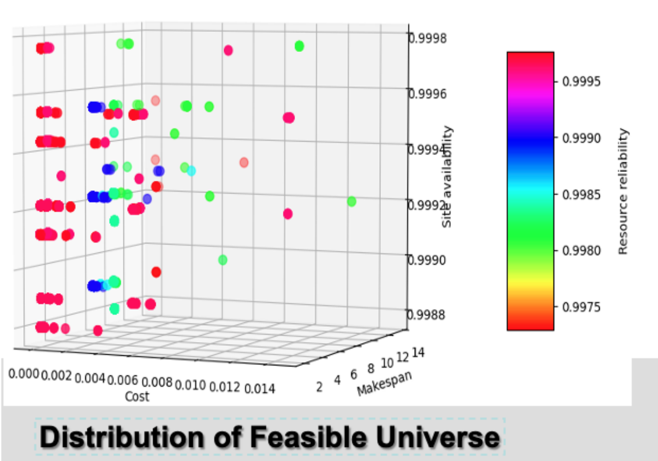
[Cloud resource configuration C]
Deploy **c4.4xlarge** in Virginia region for TopHat2
Deploy **c4.4xlarge** in Virginia region for Cufflinks
Deploy **m4.xlarge** in Virginia region for StringTie
[Execution time for genome analysis]
13440.94 sec
[Cloud resource usage cost]
4.98 USD

MOPLS Framework for Optimal Resource Selection



Many-objective Optimization via MOPLS





(Sample values)

Cost (US$)	Makespan	Availability	Reliability	Configuration
4.84060	1.89575	0.99922	0.99965	[tophat2, m4.4xlarge, Virginia], [cufflinks, c4.2xlarge, Virginia], [stringtie, m4.10xlarge, Virginia]
6.22217	1.89575	0.99944	0.99965	[tophat2, m4.4xlarge, Virginia], [cufflinks, c4.2xlarge, Virginia], [stringtie, c4.xlarge, Virginia]
8.01104	1.95569	0.99976	0.99976	[tophat2, c4.2xlarge, Virginia], [cufflinks, c4.2xlarge, Virginia], [stringtie, c4.xlarge, Virginia]
1.04113	1.89575	0.99880	0.99965	[tophat2, m4.4xlarge, Virginia], [cufflinks, m4.2xlarge, Virginia], [stringtie, r4.xlarge, Virginia]
1.09070	1.89575	0.99912	0.99965	[tophat2, m4.4xlarge, Virginia], [cufflinks, c4.2xlarge, Virginia], [stringtie, r4.xlarge, Virginia]
~	~	~	~	~

Constraint Specification Via PLS Descriptive System

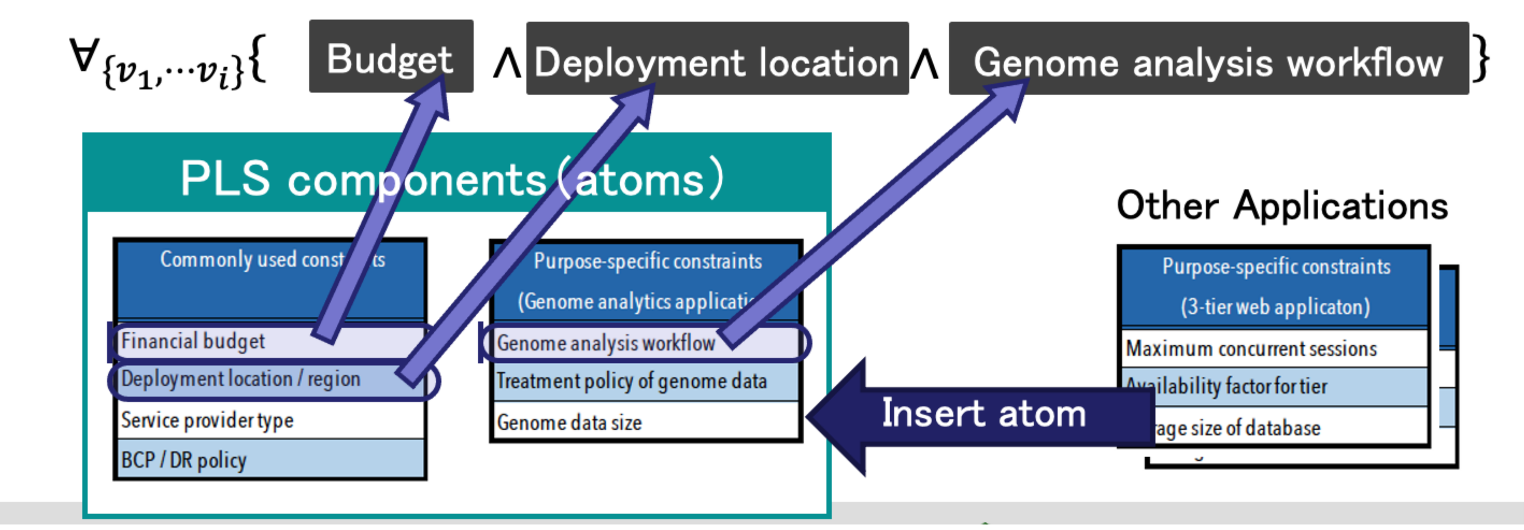
- PLS (Predicate Logic-defined Specification) is a unique formula for specifying system requirements based on first-order predicate logic.

$$\forall_{\{v_1,\cdots v_i\}}\{E_1 \wedge \cdots E_n\}$$

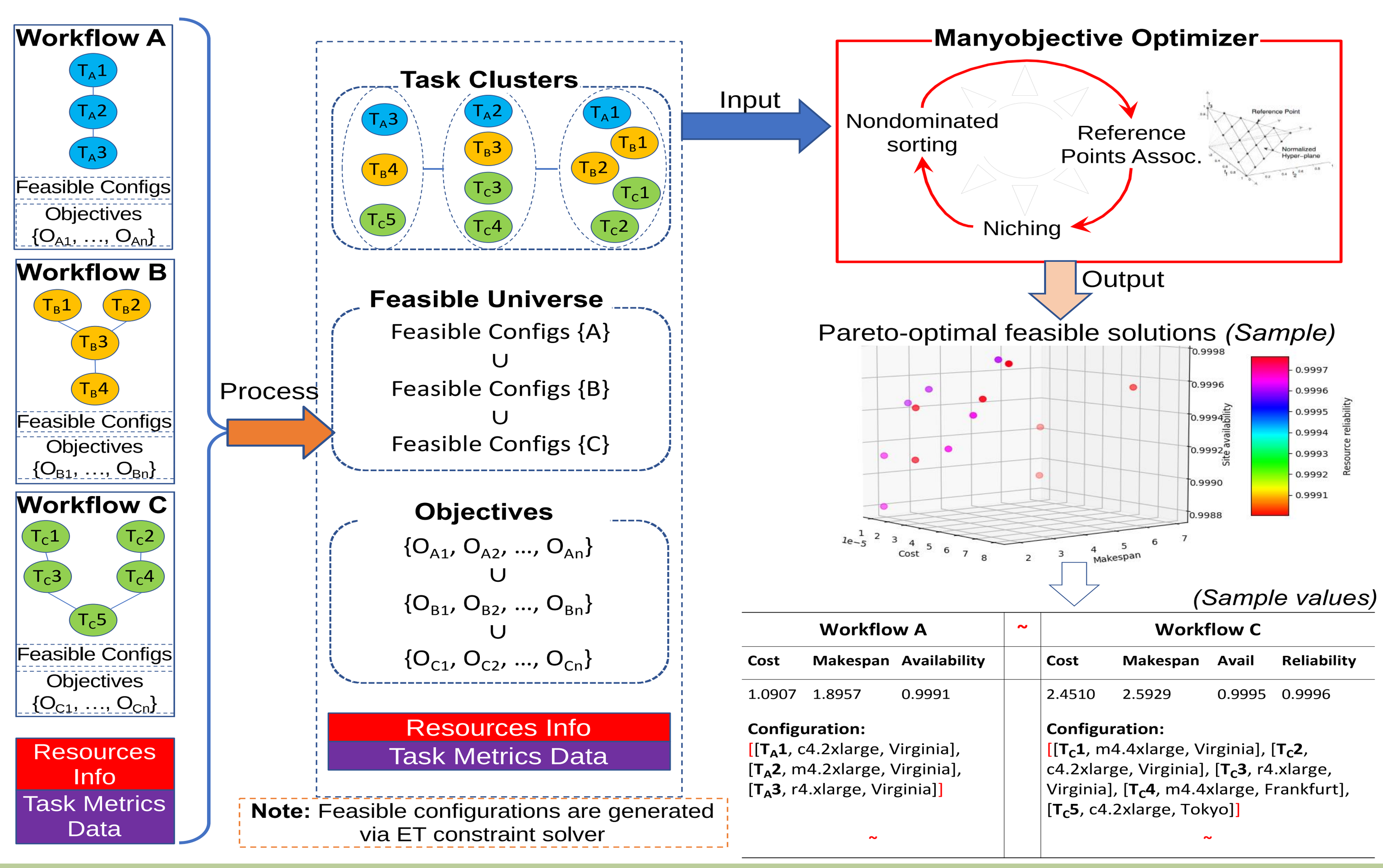
E is an atomic formula, v is a variable that appears in the atom.

- PLS is created by selecting and adding atoms required to represent system requirements

$$\forall_{\{v_1,\cdots v_i\}}\{\ \boxed{\text{Budget}} \wedge \boxed{\text{Deployment location}} \wedge \boxed{\text{Genome analysis workflow}}\ \}$$



Optimal Resource Selection for Simultaneous Workflows



Pareto-optimal feasible solutions *(Sample)*

(Sample values)

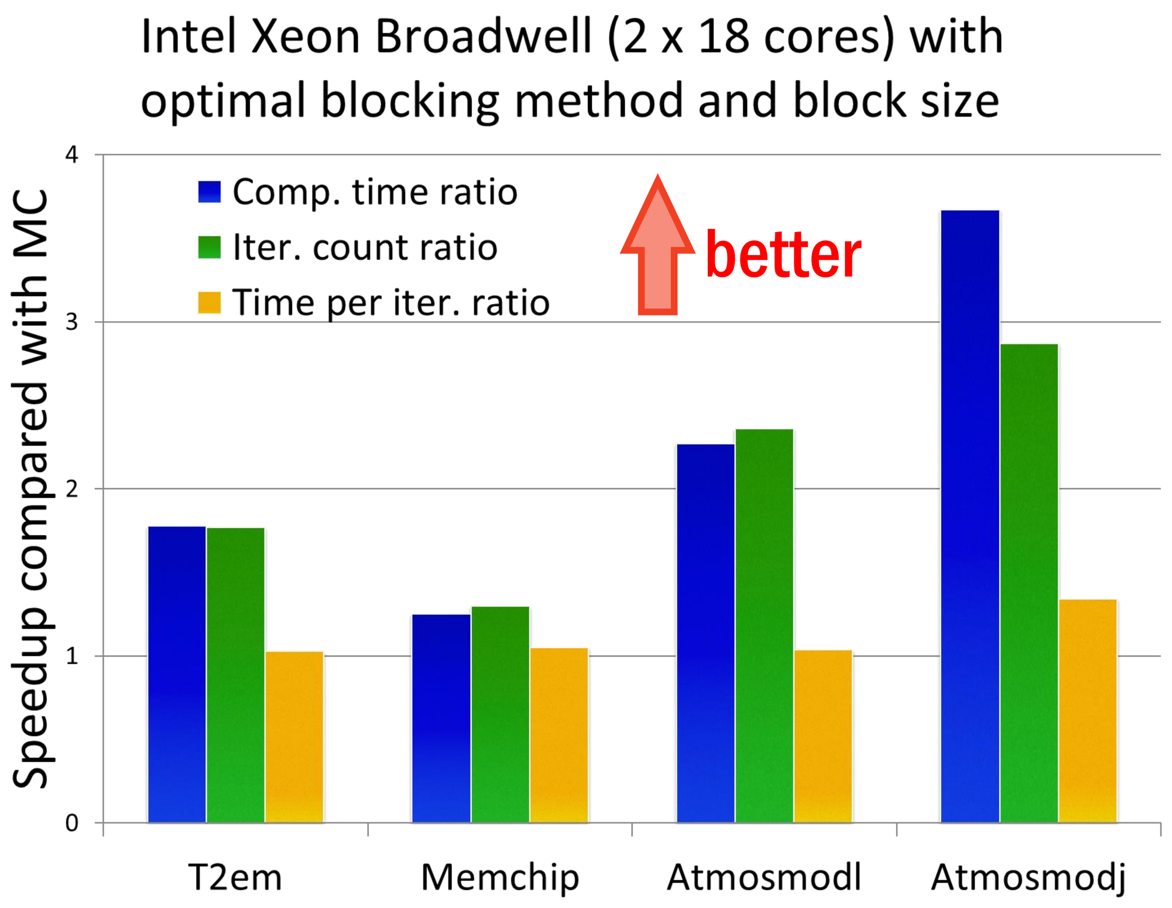
Workflow A			~	Workflow C			
Cost	Makespan	Availability		Cost	Makespan	Avail	Reliability
1.0907	1.8957	0.9991		2.4510	2.5929	0.9995	0.9996
Configuration: [[T_A1, c4.2xlarge, Virginia], [T_A2, m4.2xlarge, Virginia], [T_A3, r4.xlarge, Virginia]]				**Configuration:** [[T_C1, m4.4xlarge, Virginia], [T_C2, c4.2xlarge, Virginia], [T_C3, r4.xlarge, Virginia], [T_C4, m4.4xlarge, Frankfurt], [T_C5, c4.2xlarge, Tokyo]]			
~				~			

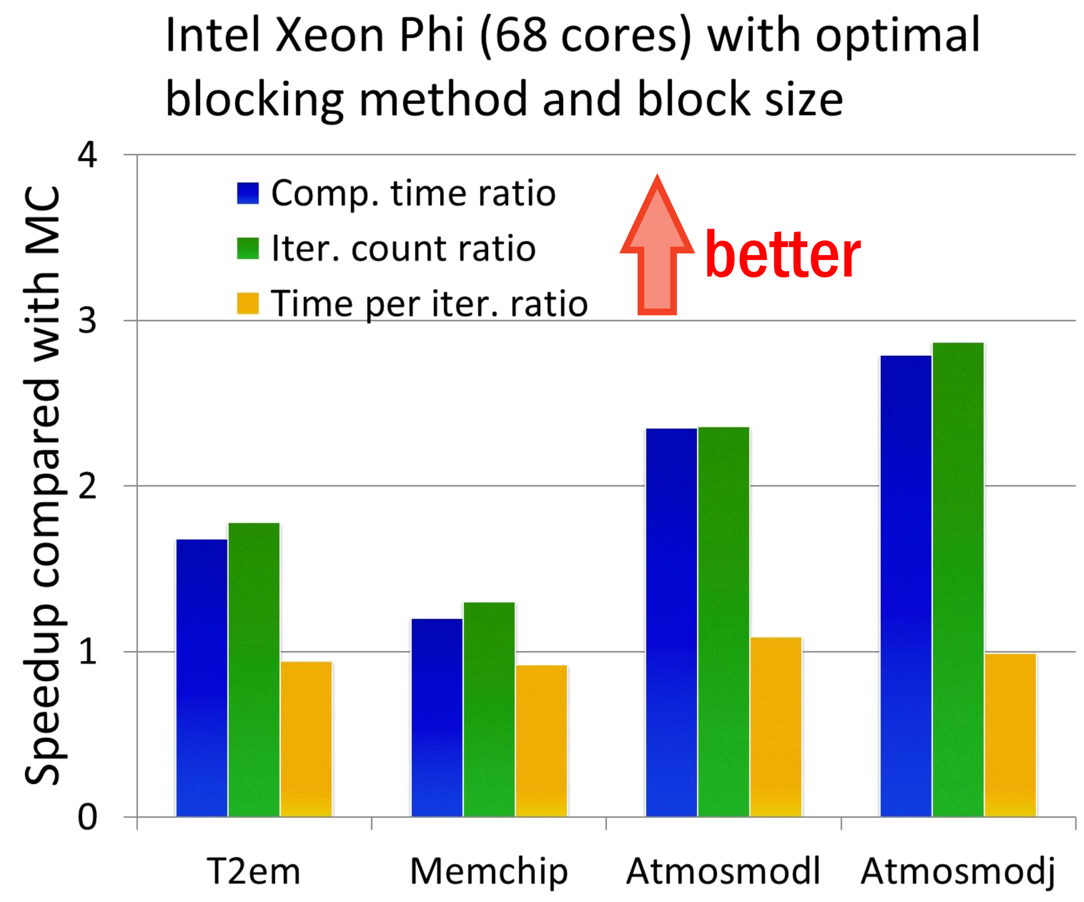
SC20 Everywhere we are | more than hpc.
HOKKAIDO UNIVERSITY
Information Initiative Center
North 11, West 5, Sapporo, Japan
Tel, Fax: +81-11-706-2923
https://www.iic.hokudai.ac.jp/en/

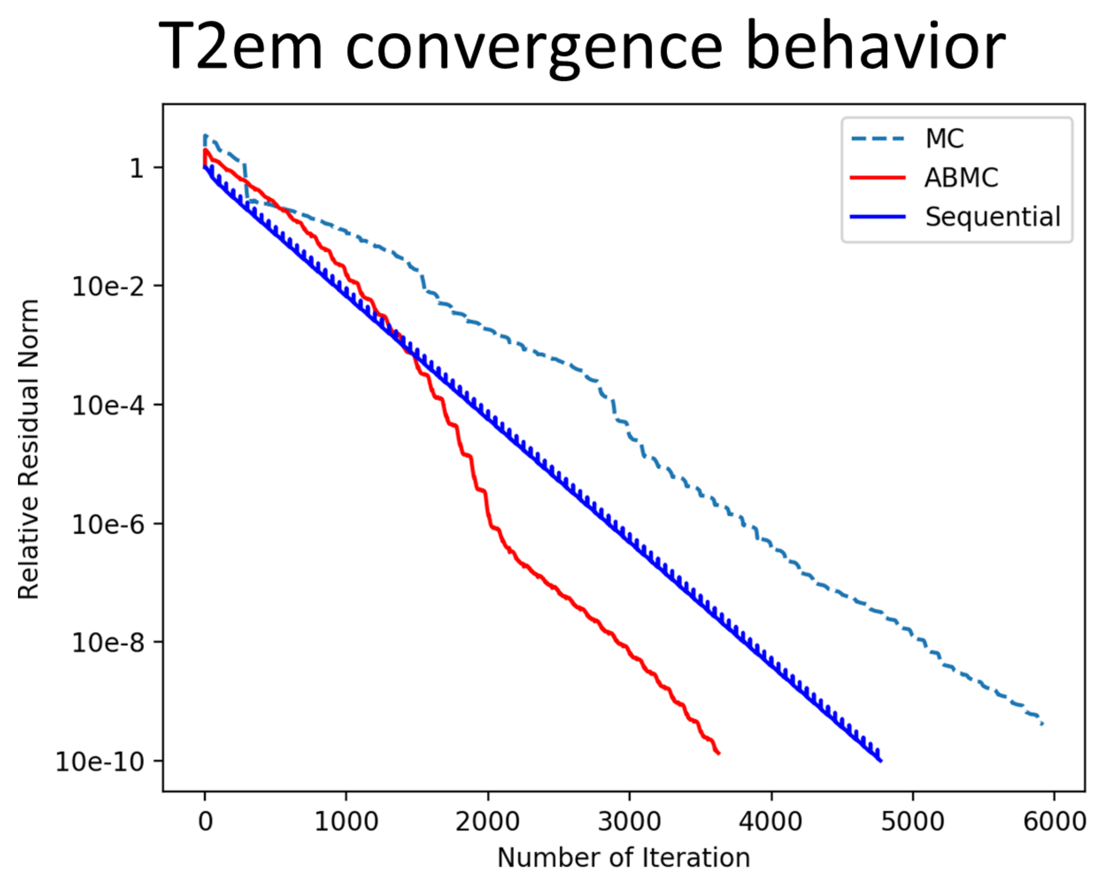
Research on High-Performance Computing

Algebraic block multi-color ordering for multi-threaded ILU preconditioner

- ILU preconditioner is one of important technique for solving unsymmetric linear system of equations by iterative solvers (e.g. GMRES solver).
- Parallelization of the triangular solver involved in ILU preconditioner is crucial to achieve high performance on a recent multi/many-core CPU.
- Algebraic block multi-color ordering is a parallel ordering technique, which achieves both fast convergence and sufficiently high degree of parallelism in parallel ICCG solver (Iwashita et al., IPDPS'12, SISC'05, and widely employed in HPCG benchmark).
- In this research, we enhance the technique to ILU preconditioner and develop a high performance multi-treaded ILU-GMRES solver.







Reference: S. Li, T. Iwashita, T. Fukaya, "Enhancement of Algebraic Block Multi-Color Ordering for ILU Preconditioning and Its Performance Evaluation in Preconditioned GMRES Solver", Journal of Information Processing, Vol. 27 (2019), pp. 201-210.

High-performance QR factorization based on the Cholesky QR algorithm

- Cholesky QR computes QR factorization by 1) $W := A^\top A$, 2) $R := \mathrm{chol}(W)$, 3) $Q := AR^{-1}$.
- Cholesky QR is suitable for high-performance computing but numerically unstable.
- We improved the stability of Cholesky QR by reorthogonalization (CholeskyQR2), but still face breakdown for ill-conditioned matrices.
- Recently, we have proposed a shift technique for Cholesky QR (shifted Cholesky QR) and use it as preconditioning before CholeskyQR2.
- The resulting algorithm (shifted CholeskyQR3) is still simple and can efficiently computes the QR factorization of ill-conditioned matrices.

$10^8 < \kappa_2(A) < 10^{16}$ $\quad$ $\kappa_2(A) < 10^8$ $\quad$ $\kappa_2(A) \simeq 1$ $\quad$ orthonormal

$$A \xrightarrow{\text{shifted Cholesky QR}} A' \xrightarrow{\text{Cholesky QR}} A'' \xrightarrow{\text{Cholesky QR}} Q$$

(Cholesky QR2)

scholqr(A, s) = [Q, R] //s>0
1. $W := A^\top A$
2. $R := \mathrm{chol}(W + sI)$
3. $Q := AR^{-1}$

cholqr2(A) = [Q, R]
1. $[A', R_1] := \mathrm{CholQR}(A)$
2. $[Q, R_2] := \mathrm{CholQR}(A')$
3. $R := R_2 R_1$

Summary the of shifted CholeskyQR3 algorithm

Reference: T. Fukaya, R. Kannan, Y. Nakatsukasa, Y. Yamamoto, and Y. Yanagisawa, Shifted CholeskyQR for computing the QR factorization of ill-conditioned matrices, SIAM J. Sci. Comput., Vol. 42 (2020), pp. A477-A503.

SC20 Everywhere we are | more than hpc.

HOKKAIDO UNIVERSITY

Information Initiative Center
North 11, West 5, Sapporo, Japan
Tel, Fax: +81-11-706-2923
https://www.iic.hokudai.ac.jp/en/