TOHOKU UNIVERSITY

A Code Transformation Approach to Achieving High Performance Portability

SPPEXA Annual Plenary Meeting
Jan 25, 2016@Leibniz Supercomputer Center

Hiroyuki TAKIZAWA
(Tohoku University/JST)
Daisuke TAKAHASHI, Reiji SUDA, and Ryusuke EGAWA

Xevolver Project

How can we help legacy application migration?

It is difficult because system-specific optimizations are tightly interwoven with application codes.

- **System-aware Code Optimizations in Existing Applications**
  - **Egawa@Tohoku-U**
    - The code patterns should be refactored because they potentially (likely) degrade the performance portability across different systems.
- **System-aware Code Optimizations for "PostPeta" Systems**
  - **Suda@U-Tokyo** and **Takahashi@U-Tsukuba**
    - New optimization techniques and algorithms for future systems
      - Communication-avoiding algorithms, etc (Suda)
      - Highly-optimized implementations for GPU clusters, etc (Takahashi).
- **Representation of System-awareness**
  - **Takizawa@Tohoku-U** (PI)
    - How to separate system-awareness from application codes

New Collaboration

TOHOKU UNIVERSITY

Xevolver (JST CREST)

- **Programming tools** for performance engineering
  - Legacy code migration
  - [illegible]
  - [illegible]
  - Software maintainability …etc

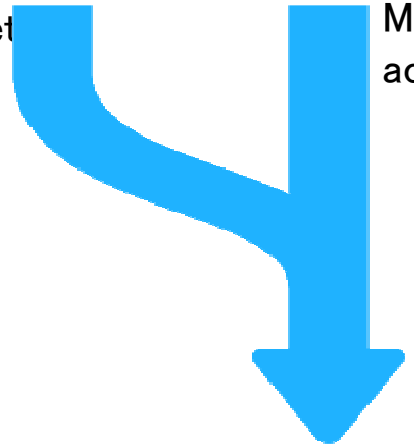
Custom code transformation

ExaFSA (SPPEXA 1st phase)

- **Exascale simulation** of FSA interactions
  - Modular multi-physics environment
  - [illegible]

Practical simulation codes (e.g. Ateles)

Multi-scale & multi-resolution fluid-acoustics … etc



ExaFSA (SPPEXA 2nd phase)

– Performance engineering in an Exascale era

Background

- HPC application development
  = team work of programmers with different concerns

Application developers (= computational scientists = ExaFSA)

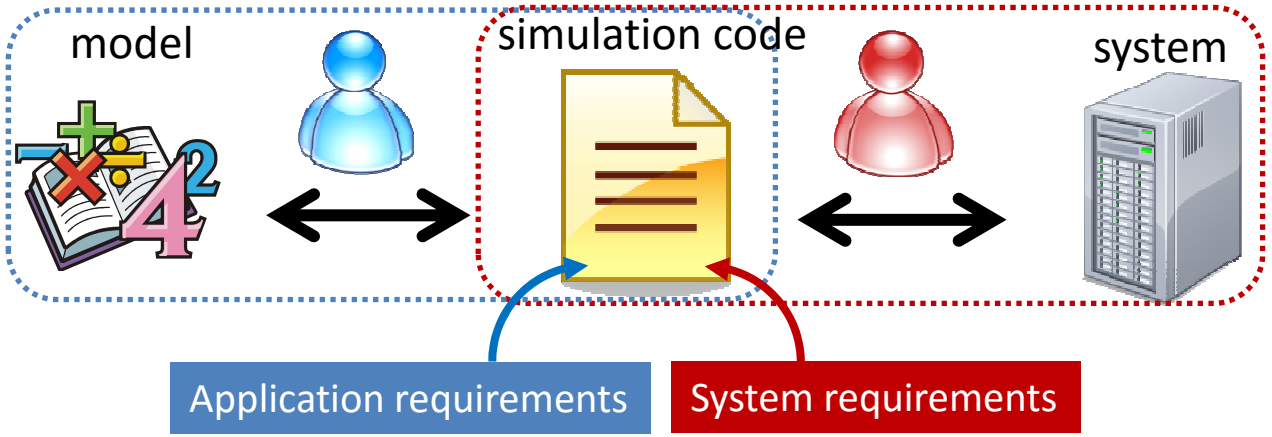
- write a program so as to get correct results

→ Main concern: relationship between simulation models and programs.

Performance tuners (= computer scientists/engineers = Xevolver)

- write a program so as to get high performance

→ Main concern: relationship between programs and computing systems.



What's the Problem?

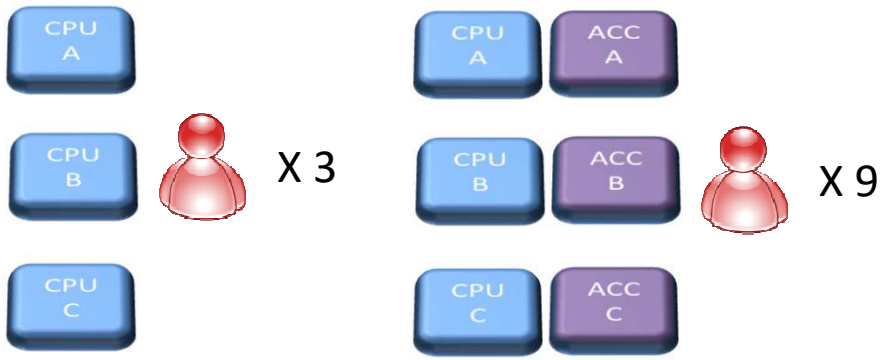
TOHOKU UNIVERSITY

- **System complexity is increasing**
  - Need to consider both parallelism and heterogeneity
  - Also need to manage deeper memory hierarchy, power, fault tolerance, …

  → System-aware optimizations are needed for high performance

  = An HPC application is specialized for a particular system

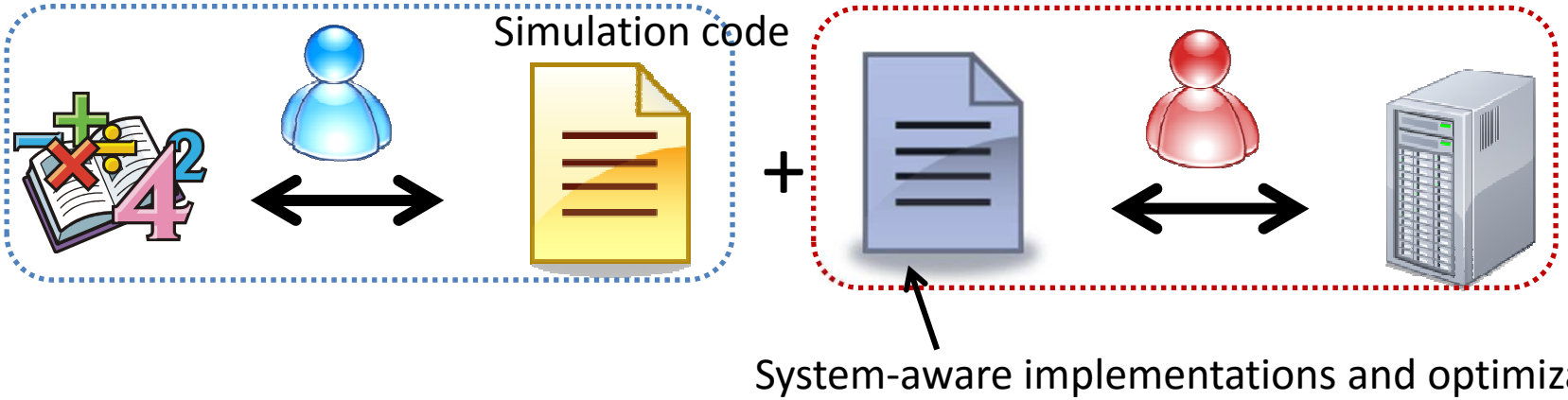
- **System diversity is also increasing**
  - Different processor combinations
  - Different system scales
  - Different interconnect network topologies
  - Different system operation policies



What can we do to achieve high performance on various systems?

Goal = Appropriate Division of Labor

Separation of system-awareness from application programs



There are many approaches to abstraction of system-awareness

- System-aware implementations with a common interface = Numerical libraries
- Standardized programming models and languages = MPI, OpenMP, OpenACC …

In reality, we still need to modify a code to achieve high performance for application-specific and/or system-specific reasons.
→ How can we abstract such code modifications?

How Is Code Modified?

TOHOKU UNIVERSITY

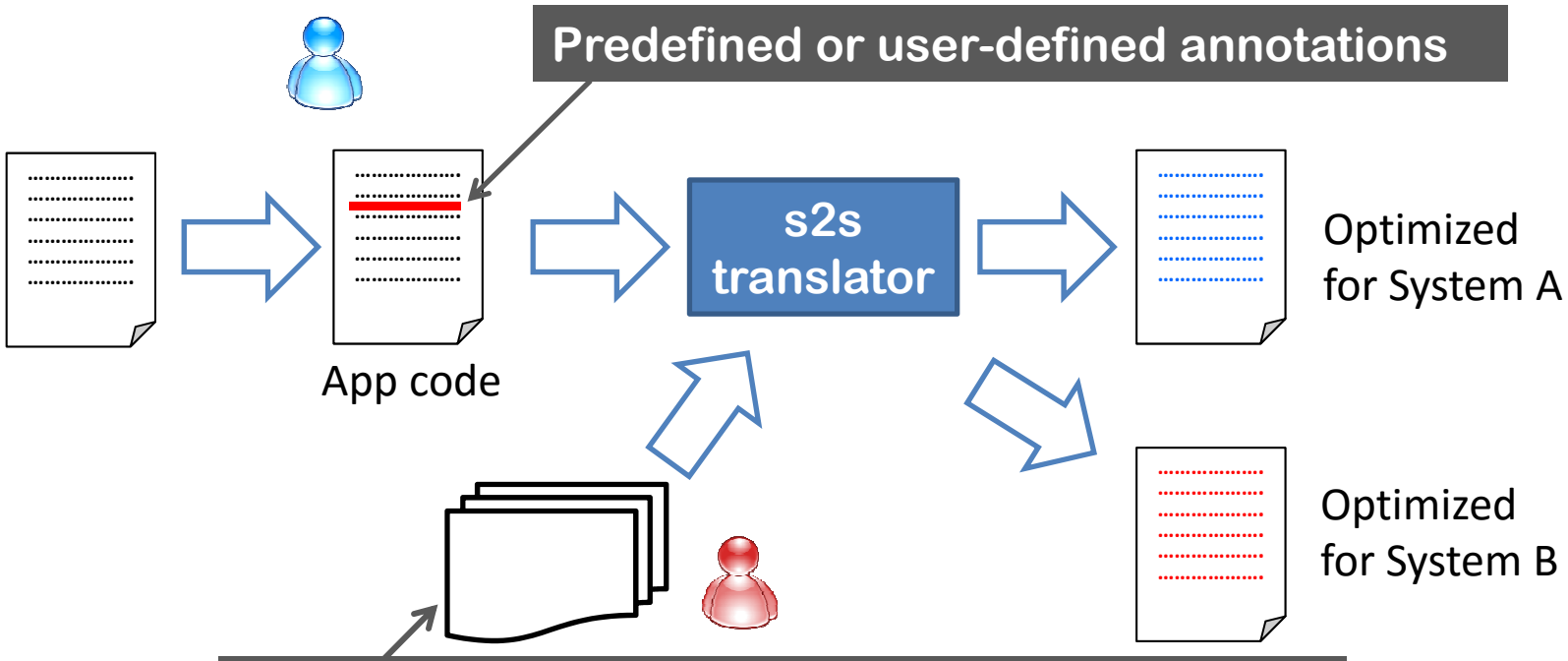
- Bad News -- Messy
  - System-aware code modifications are scattered over a code
- Good News -- Repetitive
  - Same (or similar) code modifications are required many times

Manual code modifications can be replaced with a smaller number of mechanical code transformations.

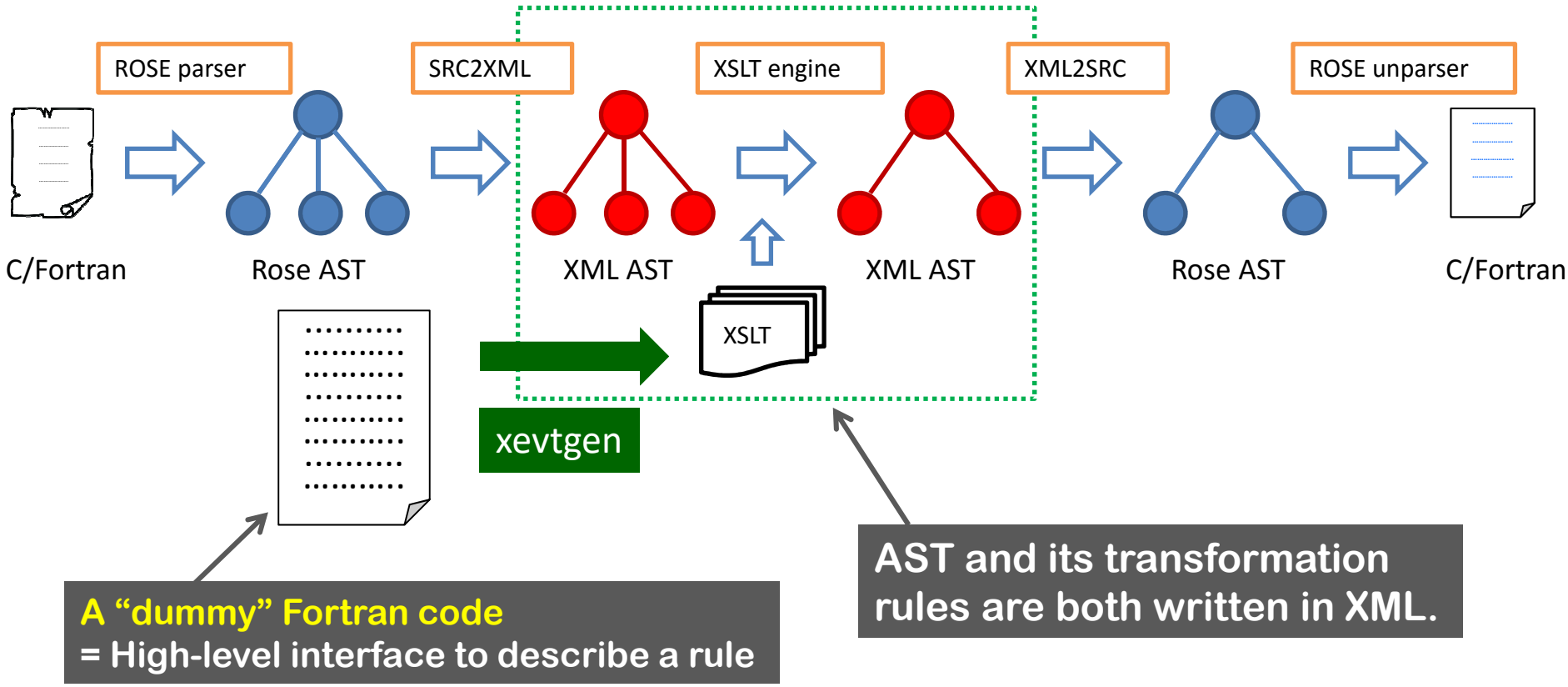
→ Express application-specific and/or system-specific code modifications as mechanical code transformations

Xevolver Framework

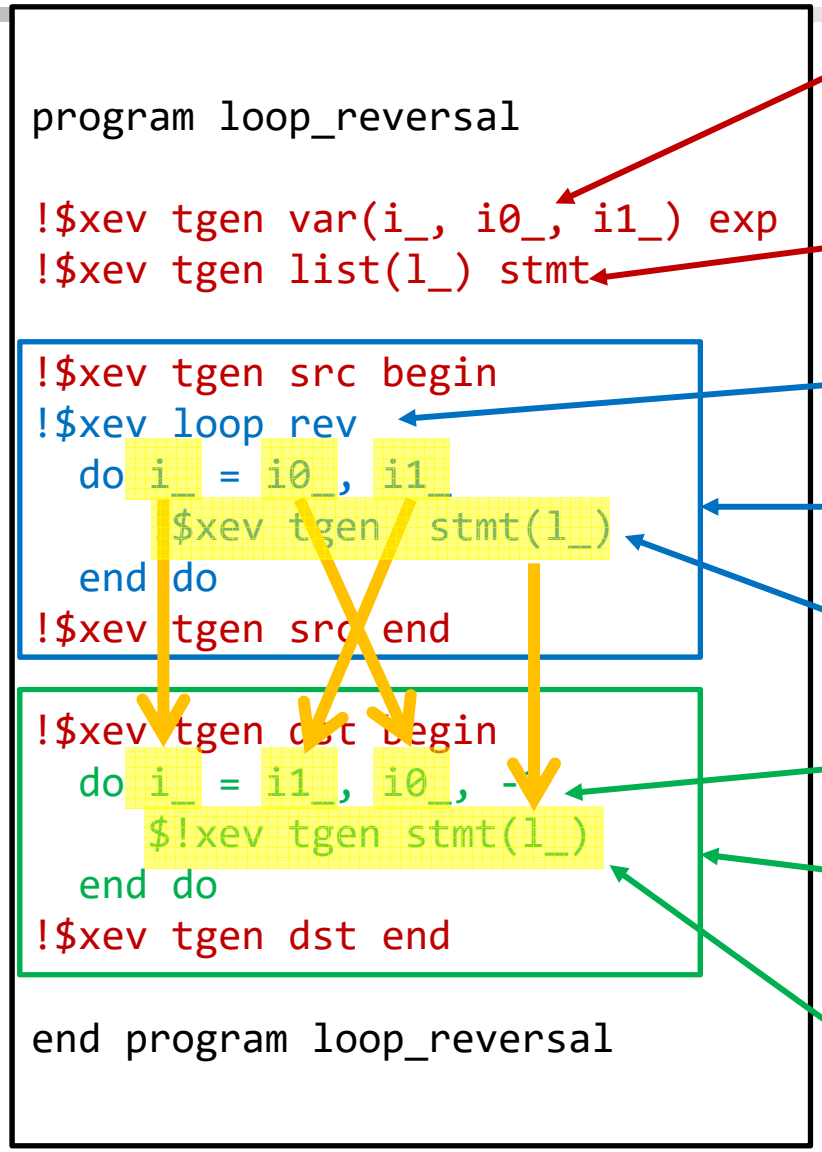
Various transformations are required for replacing arbitrary code modifications.
= cannot be expressed by combining predefined transformations.
→ **Xevolver : a framework for custom code transformations**



Xevolver and Xevtgen



Automatic Generation of Translation Rules



Tgen variables that match any expressions

A list variable that matches any number of statements

Directive used as a mark for transformation

The code pattern before transformation

Special directive that matches arbitrary statement(s)

Loop is reversed

The code pattern after transformation

The loop body is copied to the dst code.

Combining Multiple Rules

```
!$xev tgen var(i, i0, i1, p, q) exp
!$xev tgen list(body1, body2, body3) stmt

!$xev tgen src begin
!$xev loop unswitch
  do i = i0, i1
    if (p) then
      body1 = xevtgen_var
    else if (q) then
      body2 = xevtgen_var
    else
      body3 = xevtgen_var
    end if
  end do
!$xev tgen src end

!$xev tgen dst begin
  if (p) then
    do i = i0, i1
      body1 = xevtgen_var
    end do
  else
!$xev concat_if
!$xev loop unswitch
    do i = i0, i1
      if (q) then
        body2 = xevtgen_var
      else
        body3 = xevtgen_var
      end if
    end do
  end if
!$xev tgen dst end
```

```
!$xev tgen src begin
!$xev loop unswitch
  do i = i0, i1
    if (p) then
      body1 = xevtgen_var
    else
      body2 = xevtgen_var
    end if
  end do
!$xev tgen src end

!$xev tgen dst begin
  if (p) then
    do i = i0, i1
      body1 = xevtgen_var
    end do
  else
    do i = i0, i1
      body2 = xevtgen_var
  end do
end if
!$xev tgen dst end
```

```
!BEFORE
DO I=0…
  IF (…)
    …
  END IF
END DO

!AFTER
IF (…)
  DO I=0…
  …
  END DO
END IF
```

Loop Unswitching

- General rule generation with dummy Fortran + directives
- Combining two rules for supporting an arbitrary number of ELSE IF statements
- Using another rule for formatting the output code (not in this slide)

→ General rule can be defined by combining multiple simple rules.

Generated XSLT Rules

TOHOKU UNIVERSITY

Tgen can generate complicated XSLT rules from simple dummy Fortran codes.

	Tgen dummy code		XSLT template	
	Lines	Bytes	Lines	Bytes
assert	25	465	81	4,890
peel	153	3,103	856	59,034
choose	78	1,709	434	36,001
unswitch	93	1,651	503	47,822

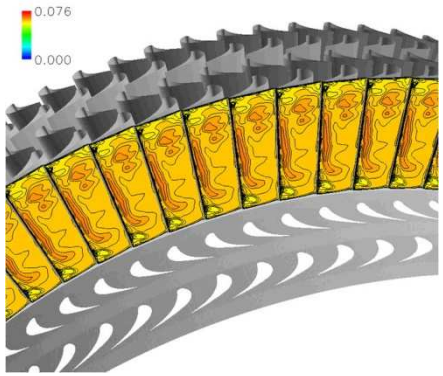
Other advantages over direct XLST writing:

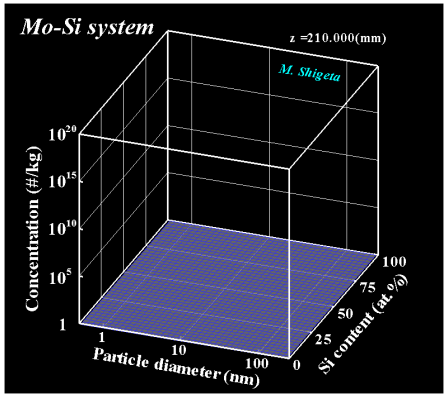
- Exact matching
- Easier priority control
- Easier to keep Fortran syntax

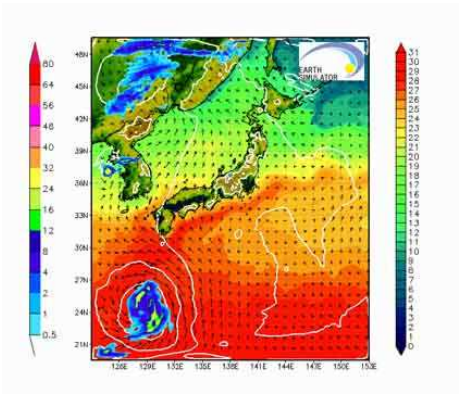
Takizawa et al@HiPC2014.

Case Studies with Real Applications

- Real-world applications originally developed for NEC SX-9 have been ported to OpenACC.
  - Numerical Turbine (Yamamoto et al@Tohoku-U)
  - Nano-Powder Growth Simulation (Shigeta@Osaka-U)
  - MSSG-A (Takahashi et al@JAMSTEC)



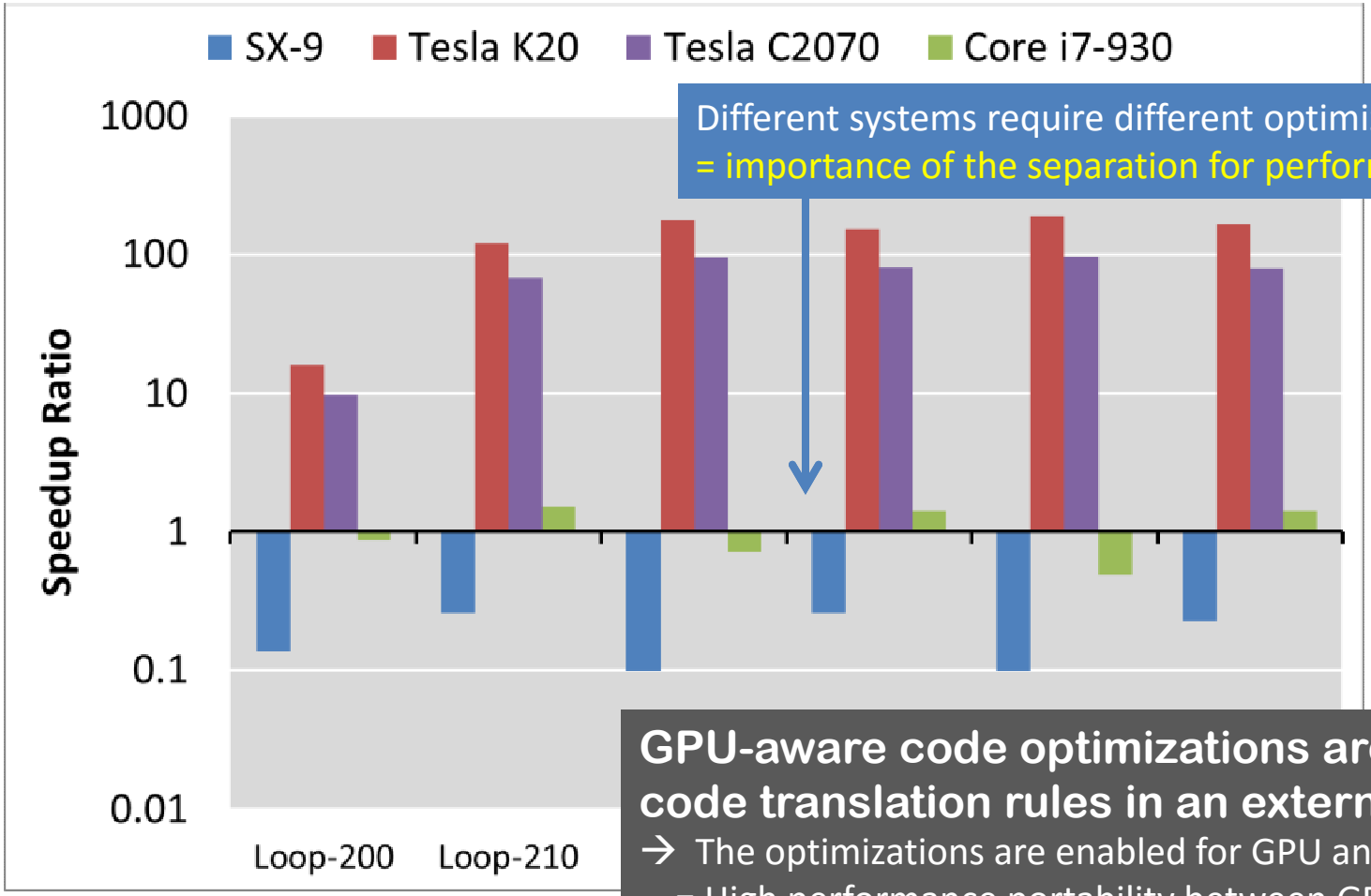




Xevolver can express system-awareness in an XML data format for migrating all the applications to OpenACC platform without major modifications.

Takizawa et al@HiPC2014.

Evaluation Results (NT)



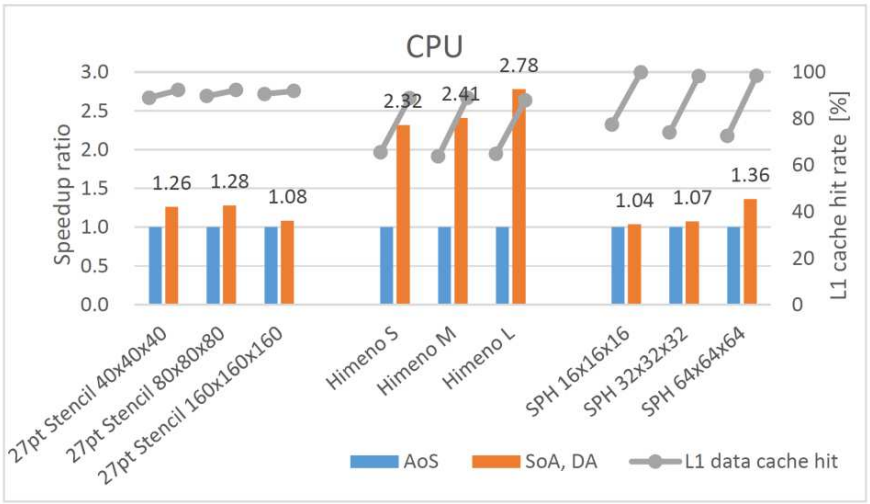
GPU-aware code optimizations are expressed as code translation rules in an external XML file.

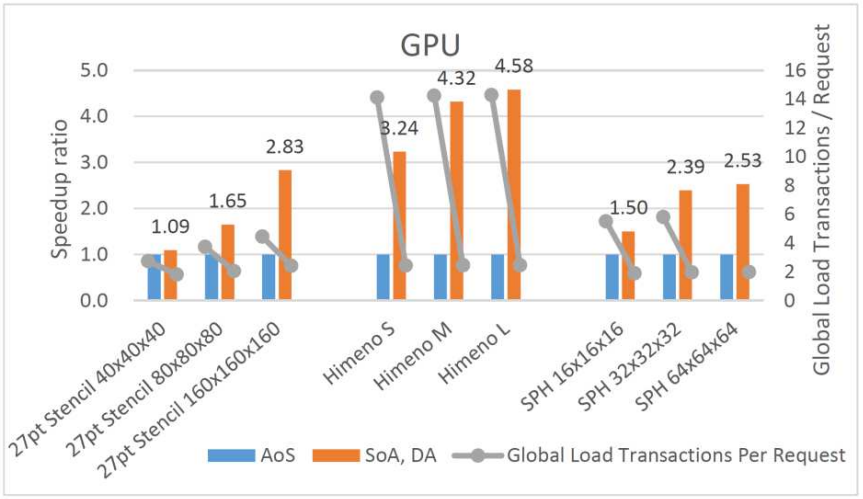
→ The optimizations are enabled for GPU and disabled for SX-9
= High performance portability between GPU and SX-9

Speedup due to code transformations (Main kernels of Numerical Turbine)

Yamada et al@LHAM2015.

Impact of Data Layout Optimization





- **Data layout optimizations can improve the performance of both CPU and GPU**
  - The GPU performance is more sensitive to the data layout.
  - The CPU performance also improves if the data size exceeds the cache capacity.
  - **The transformation rule is reusable** if customized for individual systems and applications

Conclusions

TOHOKU UNIVERSITY

- **Xevolver framework**
  - **System-specific optimizations are separated from application codes**.
    - Application developers can maintain the original code
    - Performance tuners describe system-specific optimizations in an external file
  - **Xevtgen provides a high-level interface for standard programmers to easily define their own code transformations.**
    - System-awareness is expressed as user-defined code transformations.

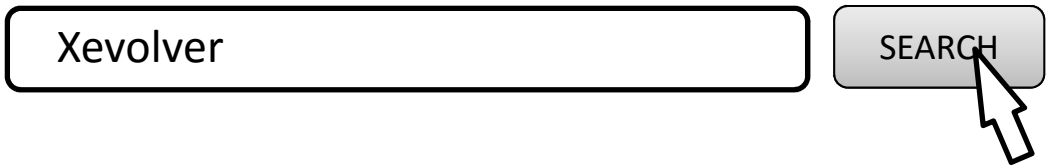
- **Future Direction**
  - We need **a standard way** to express expert knowledge and experiences in a machine-usable manner.
    - **User-defined transformations will be an important building block**

ExaFSA is a good opportunity for us to gather best practices of system-aware performance optimizations in real-world applications.

Danke!

TOHOKU UNIVERSITY

- Acknowledgements
  - This work was supported by JST Post-Peta CREST.



Xevolver with some sample translation rules is online available at http://xev.arch.is.tohoku.ac.jp.

Your feedbacks (and bug reports) are welcome!