A Tutorial Introduction to RAJA and Umpire



LLNL-PRES-853131

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC



Welcome to the RAJA Umpire tutorial

- We will start by providing a brief high-level overview of RAJA and Umpire
- We will follow the README's in the tutorial exercise directory
- The tutorial contains descriptions of RAJA and Umpire features and shows how to use them
 - The RAJA code repository contains source files with exercises based on the tutorial docs that you can work through. Complete solution files are also provided if you wish to compare with your work, or if you get stuck.
 - Please don't hesitate to ask questions at any time on Webex chat or unmute yourself during this presentation.
 - We will primarily monitor Slack after the presentation.
- Our objective for today is for you to learn enough about RAJA and Umpire to start using it in your own code development
- RAJA contains other more advanced features that will be covered in future versions of the tutorial



We value your feedback...

- If you have comments, questions, or suggestions, please let us know
 - Send us a message to our project email list: <u>raja-dev@llnl.gov</u>
- We appreciate specific, concrete feedback that helps us improve RAJA and the tutorial material



RAJA is an open-source project with a growing user and contributor base



Lawrence Livermore National Laboratory LLNL-PRES-853131

https://github.com/LLNL/RAJA



RAJA is part of the RAJA Portability Suite, which contains four complementary projects

RAJA: C++ kernel execution abstractions

Enable <u>single-source</u> application code insulated from hardware and programming model details





camp: C++ metaprogramming facilities

Focuses on HPC compiler compatibility and portability



https://github.com/LLNL/RAJA https://github.com/LLNL/CHAI https://github.com/LLNL/Umpire https://github.com/LLNL/camp



Umpire: Memory management API

High performance memory operations, such as pool allocations, with native C++, C, Fortran APIs



CHAI: C++ array abstractions

Automates data copies, based on RAJA execution contexts, giving apps the look and feel of unified memory, but with better performance



RAJA capabilities and core concepts



RAJA and performance portability

- RAJA is a library of C++ abstractions that enable you to write portable, single-source kernels that run on different hardware by re-compiling
 - Multicore CPUs, Xeon Phi, GPUs (NVIDIA, AMD, Intel), ...
- RAJA insulates application source code from hardware and programming model-specific implementation details
 - OpenMP, CUDA, HIP, SIMD vectorization, ...
- RAJA is used by many diverse production applications and libraries at LLNL and elsewhere, including ECP projects, university and vendor collaborators

The RAJA Portability Suite insulates applications from programming model and hardware architecture details





RAJA supports a variety of loop patterns and parallel constructs

Simple & complex loop patterns

- Non-perfectly nested loops
- Loop tiling
- Hierarchical parallelism
- Asychronous execution

Multiple execution back-ends

- Sequential
- SIMD (via vector intrinsics, in progress)
- OpenMP (CPU & device offload)
- Intel Threading Building Blocks (partial)
- CUDA
- AMD HIP
- SYCL (in development)

Loop patterns and transformations (without changing app code)

- Change loop iteration patterns, permute loop nest ordering
- Multi-dimensional data views with offsets and index permutations
- Hierarchical parallelism, asynchronous execution
- Direct GPU thread-block mapping control
- CPU/GPU shared and thread local memory

Portable reductions, scans, atomic operations, sorts...

Also, GPU kernel fusing (to reduce impact of GPU launch overhead for small kernels).



Simple Loops

RAJA execution policy capsulates loop execution details





Umpire capabilities and core concepts



Umpire provides a portable memory management API



Intuitive concepts

- Resources
- Allocators
- Operations

Supported memory types

- Host (CPU)
- MPI shared memory
- GPU global, constant, (host) pinned
- Unified memory
- Mmapped file memory
- Support for NVIDIA, AMD, Intel GPU devices

Features useful in HPC applications

- Various pool allocation strategies (fixed size, dynamic, monotonic, etc.)
- NUMA support
- Memory allocation advice (preferred location, mostly read, etc.)
- Thread safe allocators
- Memory introspection

- Native interfaces for C++, C, and Fortran
- Logging, backtrace, and "replay" capabilities. Useful for investigating application performance, experimenting with different allocation scenarios, finding bugs, etc.



Why We Need Umpire - Closer Look

Depending on underlying architecture being used, calls to memory can look very different:



alloc.allocate(SIZE * sizeof(float))

OpenMP Target

The same Umpire allocator can be used for several different backends.



URLs with RAJA information and examples...

- RAJA User Guide: getting started info, details about features and usage, etc. (https://readthedocs.org/projects/raja)
- RAJA Project Template: shows how to use RAJA and BLT in an application that uses CMake (https://github.com/LLNL/RAJA-project-template)
- RAJA Proxy Apps: a collection of proxy apps written using RAJA (https://github.com/LLNL/RAJAProxies)
- RAJA Performance Suite: a large collection of loop kernels used to assess compilers and RAJA performance (RAJA team, HPC vendors, DOE platform procurements, etc.) (https://github.com/LLNL/RAJAPerf)



Docs → RAJA User Guide → Getting Started With RAJA Cetit on GitHub Cetit

The RAJA project is hosted on GitHub. To get the code, clone the repository into a local working space using the command:

\$ git clone --recursive https://github.com/LLNL/RAJA.git

All of these are linked on the RAJA GitHub project page.



We maintain user documentation, tutorials, and other code repos associated with the RAJA Portability Suite projects

- Umpire User Guide: getting started info, details about features & usage, tutorial materials (readthedocs.org/projects/umpire)
- Umpire Interactive Tutorial: interactive user tutorial using Jupyter notebooks (github.com/LLNL/umpire-interactive-tutorial)
- CARE: Collection of CHAI And RAJA Externsions that are useful to application developers to help write portable code (github.com/LLNL/CARE)

🕷 Umpire develop	Docs » Umpire Cittud
Search docs BASICS	Umpire
Getting Started Umpire Tutorial	Umpire is a resource management library that allows the discovery, provision, and management of memory on next-generation hardware architectures with NUMA memory hierarchies.
REFERENCE Advanced Configuration Umpire Cookbook Features	 Take a look at our Getting Started guide for all you need to get up and running with Umpire. If you are looking for developer documentation on a particular function, check out the code documentation. Want to contribute? Take a look at our developer and contribution guides.
Doxygen	Any questions? File an issue on GitHub, or email umpire-dev@llnl.gov
CONTRIBUTING Contribution Guide Developer Guide	• Getting Started • Installation

The RAJA Performance Suite and Proxy Apps are good sources of examples for RAJA usage.

These are linked on the RAJA and Umpire GitHub projects.



Getting started with the Tutorial...





Welcome to the RADIUSS AWS Tutorial Series!

Go to:

https://software.llnl.gov/radiuss /event/2023/07/11/radiuss-onaws/

to learn more about our other tutorials and documentation!

	Date	Time (Pacific)	Project
	August 3, 2023	9:00a.m11:00a.m.	Build, link, and test large-scale applications with BLT
1	August 8–9 2023	8:00a.m11:30a.m. both days	Learn to install your software quickly with Spack
	August 10, 2023	9:00a.m11:00a.m.	Use MFEM for scalable finite element discretization application development
	August 14, 2023	9:00a.m12:00p.m.	Nuliper Integrate performance profiling capabilities into your applications with Caliper
			Analyze hierarchical performance data with Hatchet
			Optimize application performance on supercomputers with Thicket
	August 17, 2023	9:00a.m11:00a.m.	RAJV Use RAJA to run and port codes quickly across NVIDIA, AMD, and Intel GPUs
			Discover, provision, and manage HPC memory with Umpire
	August 22, 2023	9:00a.m11:00a.m.	Visualize and analyze your simulations in situ with Ascent
	August 24, 2023	9:00a.m11:00a.m.	AXCM Leverage robust, flexible software components for scientific applications with Axom
	August 29, 2023	9:00a.m11:00a.m.	Analyze runs of your code with WEAVE
	August 31, 2023	9:00a.m11:00a.m.	Flux Learn to run thousands of jobs in a workflow with Flux

Instructions for working with RAJA and Umpire on AWS

→ C 🔿 🎦 100.27.30.2	36:3000/?fold	er=/home/AWSUS	SER/r		☆			1	
EXPLORER		📔 Get Started	×						Ī
\vee raja									
> .github		< Get St	arted						
> .gitlab									
> benchmark									
> blt	S								
> build									
> cmake									
> docker			31	Get Started with VS					
> docs				Code in the Web					
> examples			- 1	Discover the best customizations					
> exercises			- 3	to make VS Code in the Web					
> host-configs				yours.					
> include		_							
> reproducers			O C	hoose the look you want					
> scripts			т	ne right color palette helps you					
> share			fo	cus on your code, is easy on your					
> src			e)	ves, and is simply more fun to					
> test			us						
> tpl				Browse Color Themes					
> travis-data			TI	p: Use keyboard shortcut (IKK IET)					
sclang-format		_							
			0 0	ne shortcut to access everything					
 gitignore 									
😝 .gitlab-ci.yml									
 .gitmodules 		0) Ji	ust the right amount of UI					
! .readthedocs.yml									
! .travis.yml				ch support for all your languages					
 () .uberenv_config.json 		,	0 10	on support for an your languages					
! appveyor.yml									
! azure-pipelines.yml		(QQ	uickly navigate between your files					
M CMakeLists.txt									
CODE_OF_CONDUCT.md			JN	lark Done Next Section →					
! codecov.yml									
R CONTRIBUTING.md									
 Dockerfile 									
* LICENSE									
■ NOTICE									
 README.md 									
= RELEASE									
RELEASE_NOTES.md									
> OUTLINE									
> TIMELINE									

- Go to the link for your VSCode environment in your email.
- VSCode is not Visual Studios, it is just an interactive text editor.
- We will still build and run our exercises from the terminal.
- VSCode has a built-in terminal.



→ C O A 100	.27.30.236:3000/?fold	er=/home/AWSUSE	R/raja	ជ	◙	Ŧ	2	🕺 🚥	0
File	> ···	■ Get Started >							
Edit	>	(Cat Star	tead						
Selection	>	C Get Star	teu						
View	>								
Go	> s								
Run	>								
Terminal	> New Terminal	^ 0 `							
Help	> Split Terminal	36/	Get Started with VS						
/ 4000	Due Teels		Code in the Web						
> examples	Run Task		Discover the best customizations to make VS Code in the Web						
> exercises	Run Build Task.	O#B	yours.						
> nost-contrigs	Run Active File								
> reproducers	Run Selected T	ext	Choose the look you want						
> scripts			The right color polatic being you						
> share			focus on your code, is easy on your						
> src			eyes, and is simply more fun to						
> test			use.						
> tpl	Configure Task	s	Browse Color Themes						
> travis-data	Configure Defa	ult Build Task	Tip: Use keyboard shortcut (%K %T)						
 .clang-format dealerionare 									
aitianore		0	One shortcut to access everything						
U.gitlab-ci.vml									
 .gitmodules 		0	Just the right amount of UI						
! .readthedocs.yml									
! .travis.yml		0	Dick support for all your languages						
{} .uberenv_config.jsor	1	0	non apport for an your languages						
! appveyor.yml									
! azure-pipelines.yml		0	Quickly navigate between your files						
M CMakeLists.txt	Tand								
CODE_OF_CONDUC	r.ma		Mark Done Next Section ->						
CONTRIBUTING.md									
✤ Dockerfile									
R LICENSE									
■ NOTICE									
③ README.md									
S RELEASE									
RELEASE_NOTES.m	d								
> OUTLINE									
> TIMELINE									

To open a terminal
 Go to: [±] → Terminal → New Terminal



Lawrence Livermore National Laboratory

- RAJA has been configure and pre-built for you in the build dir.
- cd build
- Executables live in
 - ~/raja-suite-tutorial/build/bin
- To run exercise one (from build)
 - ./bin/one



- Editing one.cpp
- make one
- ./bin/one





Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Scurity, LLC, and shall not be used for advertising or product endorsement purposes.

🕒 🔴 🕒 📗 Get Started — raja — OpenVSC: × 🛛	+	
\leftrightarrow \rightarrow C O $rac{1}{2}$ 100.27.30.236:3000/?fold	er=/home/AWSUSER/raja	☆ ♡ Ł Ø 💩 🖬 ৫ ≡
EXPLORER	Get Started ×	
Support S	Start C⊧ New File ℃ Clone Git Repository	Walkthroughs Get Started with VS Code in the Web Discover the best customizations to make VS Code in the Web yours. Carm the Fundamentals
> host-configs > include > reproducers > acripts > atrue > arc > test > test > trat- > trat-data	Recent You have no recent folders, open a folder to start.	Jump right into VS Gode and get an overview of the must-have features.
Clargeronnac deckerignore degignore gitiab-ci.yml gittab-ci.yml gittab-ci.yml	PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AMSUSERg9d5b2f58abda:-/raja/builds ./bin/dot-product	() bash - build + ∨ □ □ ∧ ×
! .travis.yml () .uberenv_config.json ! appveyor.yml ! azure-pipelines.yml	Exercise: vector dot product Running C-version of dot product (a, b) = 1e+86 Running RAJA sequential dot product	
M CMakeLists.txt CODE_OF_CONDUCT.md codecovyml CONTRIBUTING.md	(a, b) = 1e+06 result — PASS Running RAJA OpenHP dot product (a, b) = 0	
LICENSE NOTICE README.md REASE	result — FAIL Running RJA CUDA dot product (a, b) = 0	
RELEASE_NOTES.md VOUTLINE TIMELINE	result — FAIL DONE! AwSUSERgedSb2f58abda:~/raja/build§ [
ᢞ task/tut-reorg-aws ↔ ⊗ 0 🖄 0		Layout: U.S. 🖉 🚨

- RAJA has been configure and pre-built for you in the build dir.
- cd build
- Executables live in
 - ~/raja-suite-tutorial/build/bin
- To run exercise one (from build)
 - ./bin/one



C C	dot-product.cpp — raja — Open × +	-							
DPLORER ••• <	\leftrightarrow \rightarrow \mathbb{C} \bigcirc $\textcircled{2}$ 100.27.30.236:3000/?folder	r=/home/AWSUSER/raja 🔂 😇 速 🕼 👳	≡ ⊘ ≡						
<pre> AAA everclase > 0 dot-product.pp Jailab Jail</pre>	EXPLORER ····	C dot-product.cpp M ×	ti 🗆 …						
aphubo spitulo spit	→ RAJA	exercises > C dot-product.cpp							
<pre>glubb / glubb / g</pre>	└ > .github	115							
benchmark in a product served and a second product kernel using a MAA:ioon_parallel_for_execol build in a product servel and shalicon_reduce reduction policy type. benchmark in a product servel and shalicon_reduce reduction policy type.	> .gitlab	116 ///	And States of St						
bit s bit	> benchmark	118 ///	Married Street S						
 build cmake cmake cmake cmake cmake cmake cmake dcs dcs	> blt S	119 /// EXERCISE: Implement the dot product kernel using a RAJA::omp_parallel_for_exec	States						
<pre>cmake '''' cmake ''''' cmake ''''''''''''''''''''''''''''''''''''</pre>	> build	<pre>120 /// execution policy type and RAJA::omp_reduce reduction policy type. 121 ///</pre>							
> docker 123 RAJA::RoducesumeALA1::ong_reduce, doubles emplot(0.0); > warneles 123 RAJA::RoducesumeALA1::ong_reduce, doubles emplot(0.0); > warneles 124 NutrioralLeAL/org_exec>(MAJA::TypedRangeSegment <ints(0, (int="" [="]" i)="" n),="" th="" {<=""><th>> cmake</th><th>122</th><th>And the second second second</th></ints(0,>	> cmake	122	And the second second second						
<pre>> docs > examples > exercises > exercises > totofall.halday e data:-bialogam.polution.cpp e data:-bialogam.polution.cpp e</pre>	A docker	<pre>123 RAJA::ReduceSum<raja::onp_reduce, double=""> ompdot(0.0);</raja::onp_reduce,></pre>	Contraction of the local division of the loc						
A camples A cample	> docs		- marchanter						
<pre>b tutolalahday b tutolalahday c stomb-histogram.spution.cpp c stomb-histogram.spution.cpp dd t e emplot.get(); de t emplot.get(); de t e emplot.get(); de t e emplot.get(); de t e emplot.get(); de t e emplot.get(); de t emplot.get(); de t e emplot.get(); de t emplot.get(); de t e emplot.get(); de t</pre>	B vexamples	<pre>125 ompdot += a[i] * b[i];</pre>	-						
<pre>1 contact.initiogram.golution.cpp 6 adomic-histogram.golution.cpp 13 dot = ompiot.get(1); 14 dot = ompiot.get(1); 15 dot.initiogram.golution.cpp 13 dot = ompiot.get(1); 14 std::cont ce = "\tag { b, p} = " ex dot < std::end]; 15 dot.initiod.golution.cpp 14 dot.initiod.golution.cpp 15 dot.initiod.golution.cpp 16 kernel-matrix-transpose-local-array.golution.cpp 16 kernel-matrix</pre>	> tutorial balfday	127));	The second se						
10 def = deposit.ge(1); 11 get = deposit.ge(1); 12 def = deposit.ge(1); 131 get = deposit.ge(1); 132 def = deposit.ge(1); 133 get = deposit.ge(1); 134 get = deposit.ge(1); 135 dec.set(1); 136 dec.set(1); 137 get = deposit.ge(1); 138 get = deposit.ge(1); 139 dec.set(1); 131 get = deposit.ge(1); 134 get = deposit.ge(1); 135 dec.set(1); 136 dec.set(1); 137 get = deposit.ge(1); 138 get = deposit.ge(1); 139 get = deposit.ge(1); 131 get = deposit.ge(1); 131 get = deposit.ge(1); 134 get = deposit.ge(1); 135 get = deposit.ge(1); 136 dec.set(1); 137 get = deposit.ge(1); 138 get = deposit.ge(1); 139 get = deposit.ge(1); 1314 get = deposit.ge(1); <td< th=""><th>C atomic-bistogram solution con</th><th>128</th><th>BEDITINE AND A</th></td<>	C atomic-bistogram solution con	128	BEDITINE AND A						
M C MadeListstxt 131 b Dockrifie std::cout < "\t (a, b) = " < dot < std::endl; Image: manual std::endl; M C MadeListstxt 131 b Cockrifie std::cout < "\t (a, b) = " < dot < std::endl; Image: manual std::endl; M C MadeListstxt image: manual std::cout < "\t (a, b) = " < dot < std::endl; Image: manual std::cout < "\t (a, b) = " < dot < std::endl; M C MadeListstxt image: manual std::cout < "\t (a, b) = " < dot < std::cout < < td>image: manual std::cout < M C MadeListstxt image: manual std::cout < image: manual	C atomic-histogram.cop	129 dot = ompdot.get();							
#* Dockerfile 132 checkResult(dot, dot_ref); 134 #endif 132 checkResult(dot, dot_ref); 135 134 checkResult(dot, dot_ref); 135 134 checkResult(dot, dot_ref); 135 136 checkResult(dot, dot_ref); 136 136 checResult, dot_ref); 136 136 c	M CMakeLists.txt	<pre>131 std::cout << "\t (a, b) = " << dot << std::endl;</pre>							
<pre>d-ds-product_solution.cpp d-33</pre>		132	THE DOUGH AND A						
C dot-product.cpp M C karrel-matrix-transpose-local-array.cbul Kernel-matrix-transpose-local-array.cbul Kernel-matrix-transpose-local-arra	C dot-product_solution.cpp	<pre>133 checkResult(dot, dot_ref); 124 fendif</pre>	Contraction of the local division of the loc						
 ekarnel-matrix-transpose-dock-array.cpl kernel-matrix-transpose-dock-array.cpl kernel-matrix-transpose-dock-array.cpl kernel-matrix-transpose-tiled.cpp kernel-matrix-transpose-tiled.spl <th>G- dot-product.cpp M</th><th>135</th><th>an in pray and a second</th>	G- dot-product.cpp M	135	an in pray and a second						
 Image: Strange S	C- kernel-matrix-transpose_solution.cpp	136							
• karmal-matrix-transpose-local-array.pool. • karmal-matrix-transpose-tild cop • karmal-matrix-transpose-tild cop-readed: solution.cop • karmalintro-exected-loop-readed: solution.cop • karmalintro-matrix-transpose-col-al-array.pool. • launch-matrix-transpose-col-al-array.pool. • launch-matrix-transpose-col-al-array.pool. • launch-matrix-transpose-col-al-array.pool. • launch-matrix-transpose-tol-al-array.pool. • launch-matrix-transpo	C- kernel-matrix-transpose-local-array_sol	PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL	□ 自 へ ×						
0: kernel-matrix-transpose-tiled.cpp bit trajet tano 0: kernel-matrix-transpose-tiled.cpp kernel-matrix-transpose-tiled.cpp 0: kernel-matrix-transpose-tiled.cpp kernel-matrix-transpose-tiled.cpp 0: kernel-matrix-transpose-tiled.cpp kernel-matrix-tanspose-tiled.cpp 0: kernel-matrix-transpose-tiled.cpp kernel-matrix-tanspose-tiled.cpp 0: kernel-matrix-transpose-tiled.cpp kernel-matrix-tanspose-tiled.cpp 0: kernel-matrix-transpose-tiled.cpp kernel-matrix-tanspose-tiled.cpp 0: kernel-matrix-transpo	G- kernel-matrix-transpose-local-array.cpp	ANCIECEDODdEh24E0abda: /raia/builde make dot-product							
ekernel-matrix-transpose-tild.cpp Built target RAA ekernel-matrix-transpose-tocal-array_solut AustrERgedSb27Sbadds=-/rejs/build5 ekanch-matrix-transpose-tocal-array_solut Running RAA sequential dot product (a, b) = 1e466 Running RAA sequential dot product (b. luuch-matrix-transpose-tild.cpp Running RAA sequential dot product (b. luuch-matrix-transpose-tild.cpp Running RAA sequential dot product (b. luuch-matrix-transpose-tild.cpp Runing RAA sequential dot product (c) luuch-matrix-transpose-tild.cpp Festion to PSS	C- kernel-matrix-transpose-tiled_solution.c	Ault target Lange/around							
<pre> karnelinitro-matrix-transpose.opp karnelinitro-execptoja solution.opp karnelinitro-execeptoja solution.opp karnelininitro-execpt</pre>	C- kernel-matrix-transpose-tiled.cpp								
C: karnelintro-exceptols_solution.cpp Likkig CX executable/ki/a/dot-product C: karnelintro-exceptols_cop Built target dot-product C: karnelintro-nested-loop-reorder_solution.cpp Likkig CX executable/ki/a/dot-product C: kaunch-matrix-transpose-local-array.cpp Running RXA sequential dot product	G* kernel-matrix-transpose.cpp								
Ckernelinto-exected-loop-rearder_splut. Ckernelinto-exeted-loop-rearder_splut.	G kernelintro-execpols_solution.cpp	Linking COX executable/bin/dot-product Built target dot-product AMSUSER@3d5bf58abda://raj/build\$./bin/dot-product							
C: karministro-nested-loop-reorder_solution. Exercise: vector dot product C: karministro-nested-loop-reorder.pp Exercise: vector dot product C: launch-matrix-transpose-solution.pp Running C-version of dot product C: launch-matrix-transpose-solution.pp Running C-version of dot product C: launch-matrix-transpose-solution.pp Running C-version of dot product C: launch-matrix-transpose-lide Joption. Running C-version of dot product C: launch-matrix-transpose-lide Joption. Running RAJA Sequential dot product C: launch-matrix-transpose-lide Joption. Running RAJA Sequential dot product C: launch-matrix-transpose-lide Joption. Running RAJA Sequential dot product	C kernelintro-execpols.cpp								
C karnelinfo-nested-log-redder.cpp C launch-matrix-transpose-local-array.cpp C launch-matrix-tranay.cpp C launch-matrix-trana	C kernelintro-nested-loop-reorder_solutio								
c isunch-matrix-transpose-bolishrey.pol Function C-version of det product (a, b) = 1e+66 (a, b) = 1e+66 (b isunch-matrix-transpose-idial-array.pol (a, b) = 1e+66 (b isunch-matrix-transpose-idial-array.pol (b isunch-matrix-transpose-idial-array.pol (b isunch-matrix-transpose-idial-array.pol (c) isunch-matrix-transpose-idial-array.pol (c) isunch-matrix-transpose-idid.poly	C kernelintro-nested-loop-reorder.cpp	Exercise: vector dot product							
c isanch-matrix-transpose-tild_solution.c. (a, b) = 1 e+66	C launch-matrix-transpose_solution.cpp	Running C-version of dot product							
C launch-matrix-transpose-tilad_solution.c. C launch-matrix-transpose-tilad_solution.c. C launch-matrix-transpose-tilad_solution.c. C launch-matrix-transpose-tilad_solution.c. C launch-matrix-transpose-tilad_solution.c. C launch-matrix-transpose-tilad_solution.c. C launch-matrix-transpose-tilad_solution.c.	G launch-matrix-transpose-local-array_sol	(a, b) = 1e+86							
C launch-matrix-transpose-tiled.cpp (a, b) = 1e+06 C launch-matrix-transpose-tiled.cpp result PASS	C launch-matrix-transpose-tiled solution o	Running RAJA sequential dot product							
e launch-matrix-transpose con result PASS	C launch-matrix-transpose-tiled.cpp	(a, b) = 1e+06							
• Identifi matrix denopostopp	C launch-matrix-transpose.cpp	result PASS							
G- launchintro-execopls_solution.cpp Running RAJA OpenMP dot product	C- launchintro-execpols_solution.cpp	Running RAJA OpenMP dot product							
G- launchintro-execopls.cpp (a, b) = 1e+06	G- launchintro-execpols.cpp	(a, b) = 1e+06							
C- memoryManager.hpp result PASS	G- memoryManager.hpp	result PASS							
G- offset-layout-stencil_solution.cpp	C offset-layout-stencil_solution.cpp	Durada 2011 CIDE data and at							
G offset-layout-stencil.cpp (kunning kub. cub. oos product) (a, b) = 0	C offset-layout-stencil.cpp	(a, b) = 0							
8 C reductions solution.cpp	C reductions_solution.cpp	cocult EATI							
G radiutina con result - rall	Graductions con	result FALL							
DINELINE DURIERDOORSPEEdudo/raia/builds	203 TIMELINE	DONE!							
P task/tut-reorg-aws* ↔ ⊗ 0 ≙ 0 Ln 129, Col 22 Spaces: 2 UTF-8 LF C++ Layout: U.S. R Ω	} ^p task/tut-reorg-aws* ↔ ⊗ 0 △ 0	Ln 129, Col 22 Spaces: 2 UTF-8 LF C++ Layo	out: U.S. 🔊 🗘						

- Editing one.cpp
- make one
- ./bin/one

