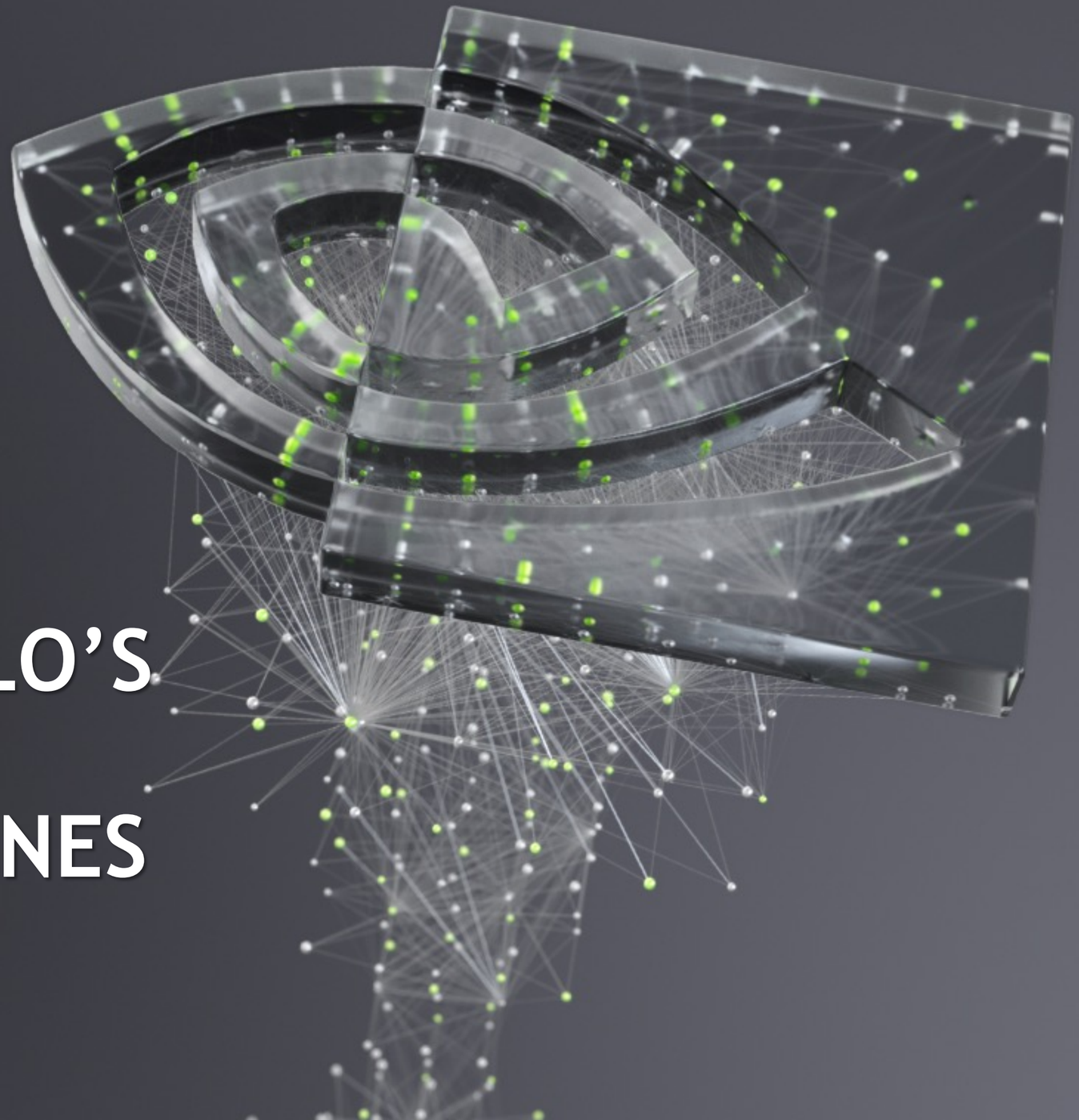
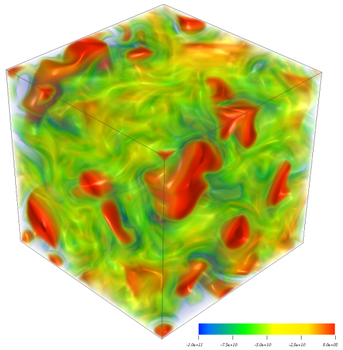


REPRISING TOMASULO'S ALGORITHM FOR DISTRIBUTED MACHINES

Michael Bauer, 10/21/25

HOW PROGRAMS RUN ON OUR DESKTOPS

Building applications is “easy”



Direct Numerical Simulation of
3D Combustion with Multi-Physics Chemistry

NumPy

Cells: 4D Arrays: [X,Y,Z,Chemical Species]
Fluxes: 5D Arrays [X,Y,Z,Chemical Species,Face]

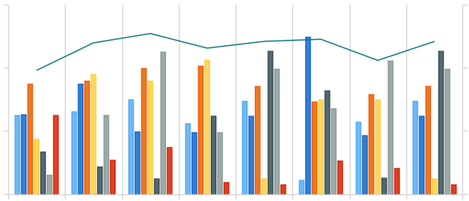
```
cells[:] += alpha * np.sum(fluxes, axis=4)
```

Uncertainty
Quantification

Pandas

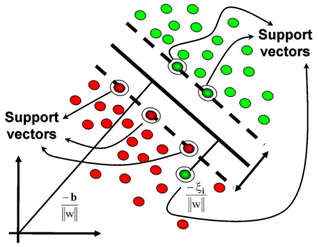
Statistical Analyses
Group-by over chemical species with different compositions

matplotlib

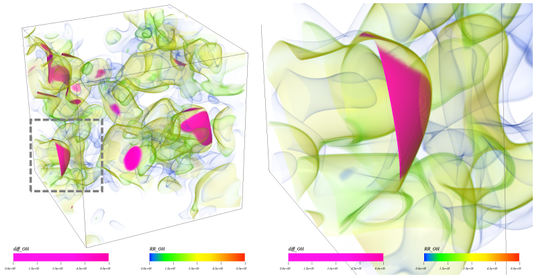


scikit learn

SVM Classifier
Recognize flame wavefronts



vtk



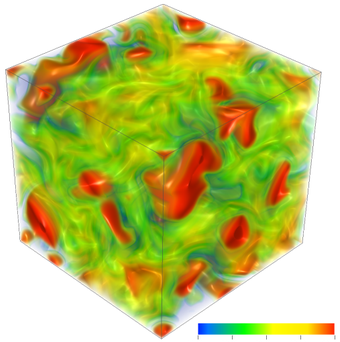
HOW PROGRAMS RUN ON OUR SUPERCOMPUTERS

Building applications is “hard”

Enter mpi4py…

Run on N nodes

Uncertainty Quantification



NumPy

Manage explicit ghost cells

```
cells[:] += alpha * np.sum(fluxes, axis=4)
```

Write/Read to/from Disk

I/O Bound

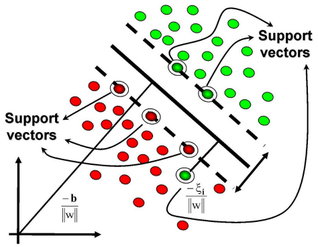
Run on M nodes

Pandas

Statistical Analyses
Group-by over chemical species with different compositions

scikit learn

SVM Classifier
Recognize flame wavefronts

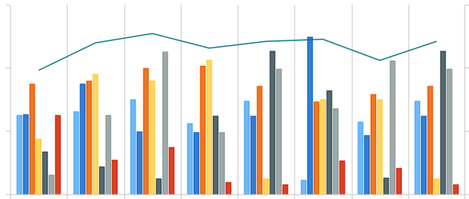


Write/Read to/from Disk

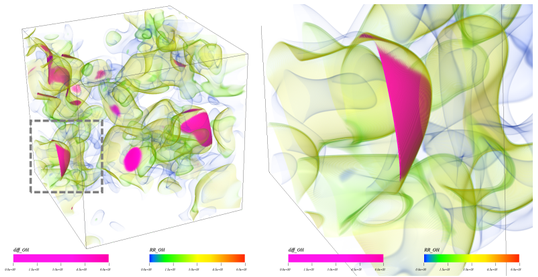
I/O Bound

Run on 1 node

matplotlib



vtk



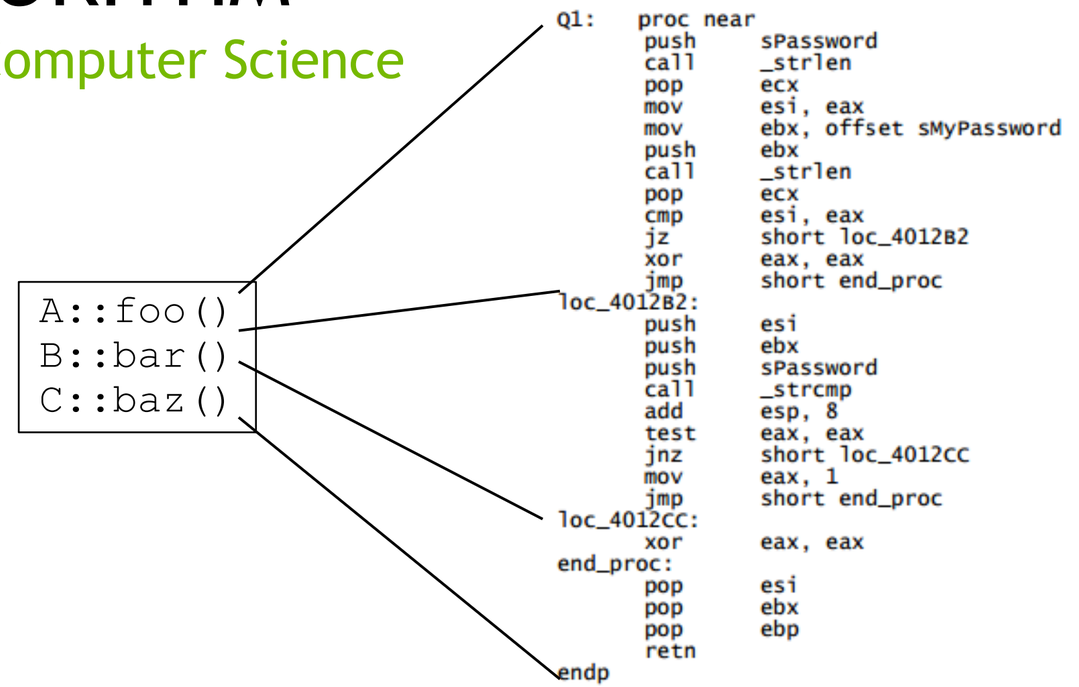
TOMASULO'S ALGORITHM

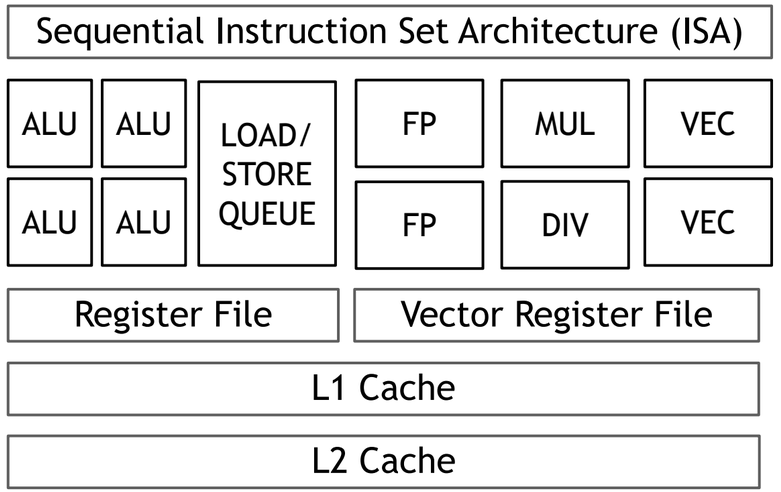
One of the Best Abstractions in Computer Science

Hardware has lots of parallel functional units and yet neither programmers nor compilers have to think about these details

Benefits:

- Sequential semantics / implicit parallelism
- Automatic latency hiding
- Coherence of data
- Composition of software
- Precise exceptions
- Portability



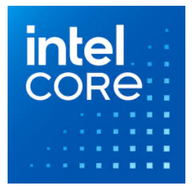


HARDWARE IS SELF-SIMILAR OVER SCALES

Why isn't software?

Increasing scale of hardware/computation

Processor



Sequential Instruction Set Architecture (ISA)

ALU ALU ALU ALU | LOAD/ STORE QUEUE | FP FP | MUL DIV | VEC VEC

Register File | Vector Register File

L1 Cache

L2 Cache

Elegant Easy-to-Use Programming Model

Sequential Semantics/Implicit Parallelism

Composable Software

Chip

???

SM SM SM SM | TMA | RT RT | MMA MMA | TEX TEX

SHARED MEMORY | TENSOR MEMORY

NOC

L2 Cache

Challenging Programming Models (MPI/CUDA/UPC/Co-Array Fortran/Global Arrays …)

Explicit Parallelism, Explicit Data Movement, Explicit Synchronization

Hardly Any Software Works Together Without Massive Effort

Supercomputer

???

CPU CPU CPU CPU | NIC(s) | GPU GPU | FPGA FPGA | ASIC ASIC

DRAM | HBMEM

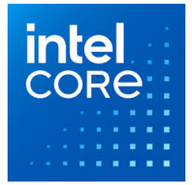
NVLink/Infiniband/Slingshot

Distributed File System with Burst Buffers

TOMASULO'S ALGORITHM STILL HAS MUCH TO OFFER

Remember the Lessons We Should Already Have Learned

Increasing scale of hardware/computation

Processor	Chip	Supercomputer
		
Sequential Instruction Set Architecture (ISA)	Implicitly Parallel Task-Based Programming Models	
Tomasulo's Algorithm in Hardware	Runtime Systems that Implement Tomasulo's Algorithm	

Hygienic and ergonomic software ecosystems demand that (almost) all parallelism should be implicit

All software should trivially compose regardless of scale

All software should be portable

Lack of discipline on our part to maintain good abstractions like those enabled by Tomasulo's algorithm

Programs Are Dynamic Sequence of Tasks

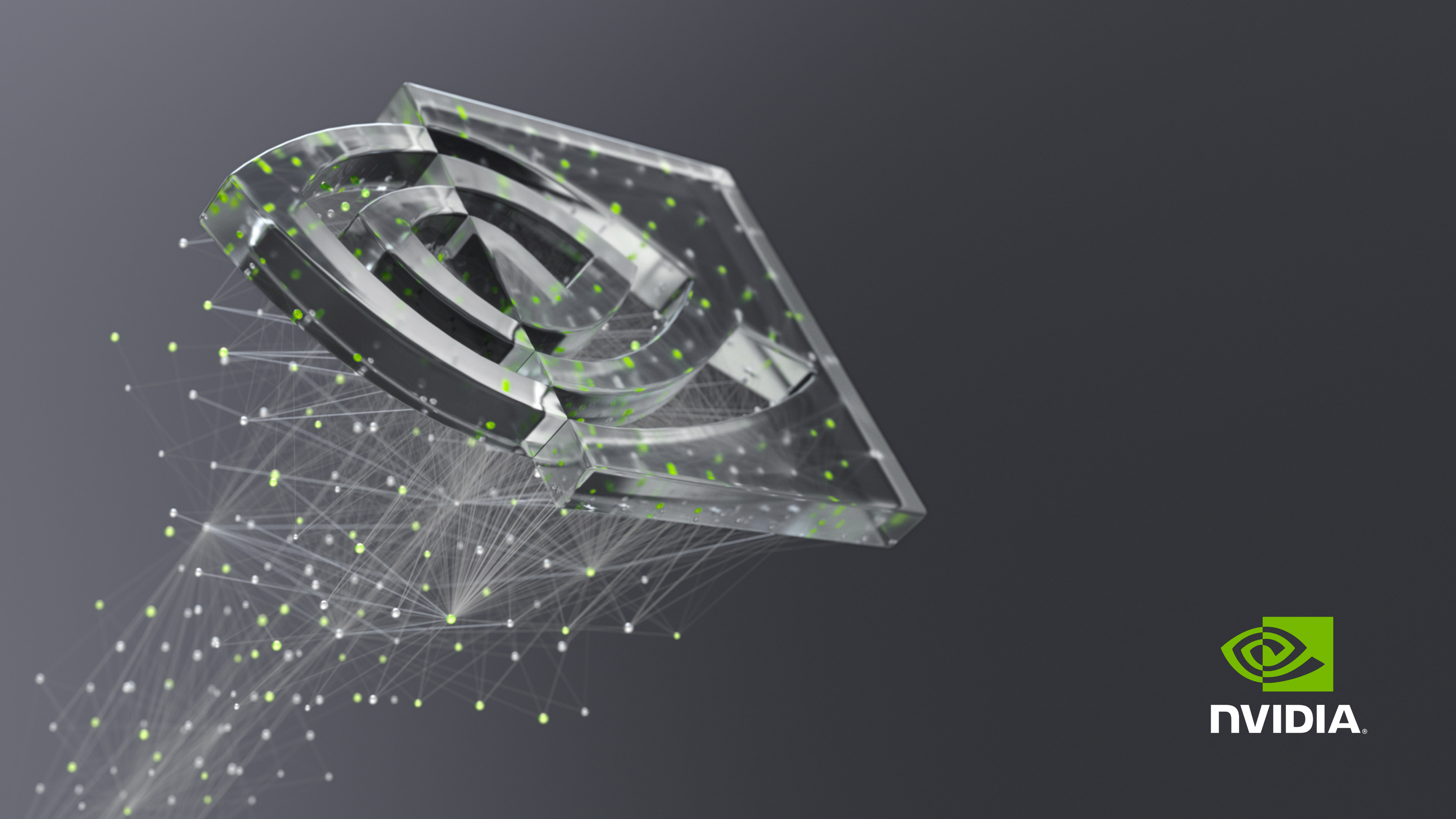
Tasks Name Region Usage

Tasks Recursively Decompose At Different Scales

t1: r1, r3, r4
t2: r2, r4
t3: r1, r2, r5, r6
t4: r3, r4, r6
t5, r2, r4, r5

ta: r1, r5
tb: r2, r5, r6
tc: r2, r6
td: r1, r2

Instructions : Registers :: Tasks : Regions


NVIDIA