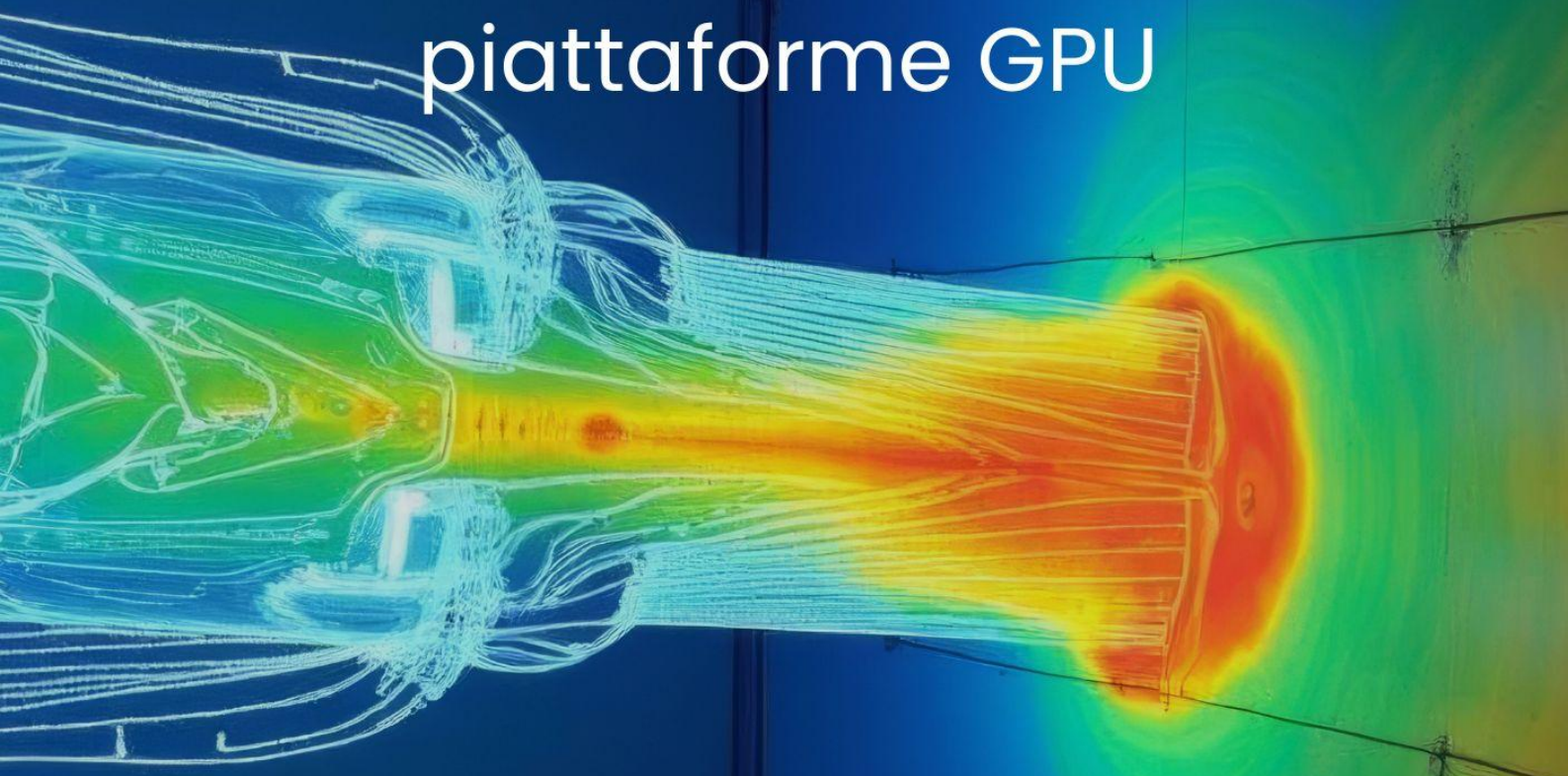
EUROCC ITALY
HPC Competence Center

# SPUMA:

## il potenziale della CFD open-source sulle piattaforme GPU



Autori: Simone Bnà, Giuseppe Gianquinto, Tommaso Zanelli, Alessio Piccolo, Francesco Bottau, Ettore Fadiga, Vittoria Sibilia, Eric Pascolo

# SPUMA il potenziale della CFD open-source sulle piattaforme GPU

*Autori: Simone Bnà, Giuseppe Gianquinto, Tommaso Zanelli, Alessio Piccolo, Francesco Bottau, Ettore Fadiga, Vittoria Sibilia, Eric Pascolo*

## Sommario

# Introduzione

Il presente white paper ha per oggetto SPUMA, un porting su acceleratori GPU, sviluppato da Cineca, del noto software open-source di fluidodinamica computazionale OPENFOAM®.[1]

I primi due capitoli offrono una panoramica su OpenFOAM e sulla sua rilevanza per le applicazioni industriali del calcolo ad alte prestazioni (HPC). I due capitoli successivi illustrano la metodologia adottata per il porting del software su GPU e i vantaggi che l'utilizzo degli acceleratori può apportare ai flussi di lavoro di simulazione. Il quinto capitolo è dedicato all'adozione della metodologia Agile nello sviluppo di SPUMA, mentre il sesto ne riporta le prestazioni nell'esecuzione di un caso di test rappresentativo di applicazioni industriali. Infine, le conclusioni offrono una panoramica dei benefici di SPUMA rispetto ai tradizionali software di simulazione basati su CPU, insieme agli effetti positivi derivanti dall'impiego dell'approccio Agile nel suo sviluppo.

[1] OPENFOAM® è un marchio registrato di OpenCFD Limited, che produce e distribuisce il software OpenFOAM. Questa offerta non è approvata o sostenuta da OpenCFD Limited, produttore e distributore del software OpenFOAM tramite www.openfoam.com e proprietario dei marchi OPENFOAM® e OpenCFD®.

# Cos’è OpenFOAM

La dinamica dei fluidi influenza tutti i campi dell'ingegneria, nonché una grande varietà di fenomeni fisici complessi. Dall’avvento del calcolo automatico, la Fluidodinamica Computazionale (o CFD, da Computational Fluid Dynamics) è diventata uno strumento essenziale per scienziati e ingegneri, per supportare i processi di progettazione e minimizzare la necessità di costosi esperimenti.

OPENFOAM® è un software CFD orientato all’HPC, sviluppato principalmente da OpenCFD Ltd a partire dal 2004. È gratuito, open-source e dispone di una vasta comunità internazionale di utenti sia nel mondo industriale che in quello accademico. OpenFOAM è uno strumento per risolvere Equazioni Differenziali alle Derivate Parziali (o PDE, da Partial Differential Equations) per applicazioni scientifiche ed ingegneristiche, spaziando dalla dinamica dei fluidi con reazioni chimiche e turbolenza alla meccanica dei solidi e all'elettromagnetismo. È stato ampiamente utilizzato in studi di aerodinamica esterna nell’industria aerospaziale e automobilistica, per valutare efficienza e coppia, nel settore delle turbomacchine e in simulazioni di impianti di condizionamento per calcolare scambi termici, portate e generazione/trasporto di umidità.

OpenFOAM utilizza il metodo dei Volumi Finiti del secondo ordine a variabili collocate per discretizzare PDE come le equazioni di Navier-Stokes: lo stesso algoritmo che si trova nei software CFD commerciali come Ansys Fluent e StarCCM+. Si basa su mesh non strutturate di celle poliedriche, che possono essere generate utilizzando strumenti nativi (come snappyHexMesh) o strumenti commerciali (come CFMesh+ e Pointwise).

# Tecnologie di High-Performance Computing (HPC) per simulazioni ad alta fedeltà

OpenFOAM può essere eseguito in parallelo utilizzando il paradigma MPI (Message Passing Interface) e tecniche di decomposizione del dominio; scala dai pochi core disponibili su un laptop o workstation fino alle decine di migliaia di core presenti su un cluster HPC, un insieme di server distinti connessi tramite una rete ad alta velocità.

Grazie alla sua natura gratuita ed open-source, OpenFOAM non comporta costi di licenza. Può essere utilizzato su cluster di High-Performance Computing (HPC) pagando solo per le risorse computazionali richieste dalla simulazione. La scalabilità parallela del codice dipende da vari fattori, alcuni algoritmici ed altri legati alle caratteristiche dell'hardware. È ragionevole assumere che un minimo di 10.000 celle computazionali per core CPU sia necessario. Queste caratteristiche rendono OpenFOAM un'alternativa economica ai software commerciali per simulazioni complesse caratterizzate da un elevato numero di celle.

La potenza computazionale disponibile per ricercatori ed ingegneri è cresciuta continuamente grazie alla recente introduzione di cluster ibridi CPU - GPU. Questo ha portato molti nel settore industriale a guardare oltre l'approccio tradizionale delle equazioni di Navier-Stokes mediate alla Reynolds (o RANS, da Reynolds-Averaged Navier-Stokes), verso simulazioni ad alta fedeltà come Large-Eddy Simulation (LES) o ibride RANS-LES per flussi di interesse industriale. OpenFOAM, tuttavia, è stato progettato negli anni Novanta, quando l'hardware, diversamente da oggi, consisteva in configurazioni a singolo core CPU con quantità molto inferiori di memoria RAM e senza accelerazione GPU per operazioni matematiche.

Le CPU sono unità di calcolo *general-purpose*, implementate su microprocessori a circuiti integrati, storicamente utilizzate per il calcolo scientifico, oltre a numerose altre applicazioni. Solo di recente le GPU hanno cominciato ad essere sfruttate per accelerare le parti più

numericamente costose delle simulazioni. L'architettura di una GPU differisce da quella di una CPU poiché originariamente era stata progettata per accelerare l'elaborazione delle immagini digitali e la grafica informatica. L'esecuzione di software sviluppato per CPU sulle GPU richiede un'attività di adattamento o *"porting"* alla nuova architettura: ciò può essere un compito impegnativo se il codice consiste di milioni di righe come OpenFOAM.

Al giorno d'oggi, le GPU forniscono la maggior parte della potenza computazionale di un cluster, misurata in FLOPS (Floating Point Operations Per Second). In quanto tecnologia cruciale per lo sviluppo dell'intelligenza artificiale, le GPU stanno evolvendo rapidamente. Esse dispongono di una larghezza di banda della memoria elevata (nell'ordine dei terabyte al secondo), una caratteristica particolarmente rilevante per applicazioni limitate dalla velocità della memoria *(memory-bound)* come OpenFOAM. La possibilità di eseguire OpenFOAM sulle GPU può avere un impatto significativo sul settore industriale: simulazioni ad alta fedeltà potrebbero essere eseguite utilizzando un numero limitato di GPU ed impiegando lo stesso tempo di una simulazione a bassa fedeltà eseguita su un numero considerevole di core CPU.

# SPUMA: la prima versione *production-ready* di OpenFOAM eseguibile su GPU

SPUMA (Simulation Processing on Unified Memory Accelerators) è un fork di OpenFOAM rilasciato da CINECA che permette al software di essere eseguito su GPU.

Negli ultimi anni, i fornitori di software CFD commerciale hanno iniziato a sviluppare versioni dei propri prodotti eseguibili su GPU. C'è un notevole interesse sia da parte degli sviluppatori che degli utenti di software di simulazione per i significativi guadagni di prestazioni che hardware accelerati come le GPU possono offrire. Questi lavori di *porting* da parte dei fornitori di software spesso comportano una riscrittura completa del loro codice, per adattarlo in maniera ottimale all'hardware in questione. Tuttavia, la comunità open-source che mantiene OpenFOAM non dispone di una quantità equivalente di risorse da dedicare a un'attività di tale portata.

Il successo di OpenFOAM può essere attribuito alla natura modulare della sua interfaccia, o *frontend:* si tratta di una sintassi di alto livello per tradurre sistemi di equazioni differenziali alle derivate parziali in solutori eseguibili, consentendo la simulazione di una vasta gamma di fenomeni fisici. D'altro canto, la sua popolarità come software open-source è dovuto anche alla rigidità del suo *backend:* le routine numeriche di basso livello vengono cambiate raramente, riducendo lo sforzo di manutenzione richiesto dal codice.

Nel corso dell'ultimo decennio, sono stati fatti diversi tentativi di realizzare un *porting* di OpenFOAM su GPU, con vari gradi di successo. Questi tentativi non prevedevano una riscrittura completa del software, ma si concentravano piuttosto o sul porting di alcune parti del codice particolarmente compute-intensive o sull'integrazione con framework già esistenti per l'accelerazione su GPU (ad esempio Thrust, Kokkos). Il primo approccio si è concentrato soprattutto sulla risoluzione dei sistemi di equazioni lineari utilizzando librerie di algebra lineare sviluppate per GPU.

Poiché l'integrazione di queste librerie poteva essere contenuta in plugin indipendenti, è risultata accettabile dalla comunità di OpenFOAM; tuttavia, i guadagni di prestazioni ottenibili attraverso questo metodo sono limitati. Il secondo approccio, sebbene più promettente in termini di prestazioni, può incidere in modo significativo sul backend a basso livello del codice a causa dell'introduzione di chiamate alle API. Per questo motivo, nessuno dei tentativi di porting basati su framework esistenti è mai stato incluso nelle release ufficiali di OpenFOAM e, spesso, i piccoli team che li sviluppano non dispongono delle risorse necessarie per mantenerli come fork indipendenti del codice.

CINECA ha cominciato a lavorare ad un *porting* di OpenFOAM su GPU nell'ambito del progetto exaFOAM, finanziato dall'UE, che è durato dal 2021 al 2024. L'obiettivo del progetto era quello di individuare modalità per aumentare le prestazioni di OpenFOAM sulle moderne piattaforme di calcolo ad alte prestazioni ibride.

Il contributo di CINECA al progetto è stato un proof-of-concept, una versione parziale di OpenFOAM compatibile con GPU chiamata zeptoFOAM, che ha dimostrato come sia possibile ottenere notevoli incrementi di prestazioni senza modificare in modo significativo gli algoritmi di alto livello e le strutture dati del codice. Grazie all'esperienza maturata nello sviluppo di questo prototipo e all'analisi dei limiti dei precedenti tentativi di porting, CINECA ha individuato i seguenti requisiti per un porting completo e di successo dell'intero codice:

- **Impatto minimo sul codice:** modifiche radicali a OpenFOAM hanno minori probabilità di essere accettate dalla comunità che ne mantiene il codice, e rendono più difficile la manutenzione di un eventuale fork.
- **Nessuna dipendenza da framework esterni:** oltre ai problemi già citati, i framework di terze parti in genere non consentono di ottenere gli stessi miglioramenti di prestazioni ottenuti con le API di basso livello.

- **Gestione efficiente della memoria:** OpenFOAM crea continuamente strutture dati di breve durata, e ciò impatta le GPU maggiormente rispetto alle architetture tradizionali a causa del costo più elevato per l'allocazione della memoria. Questo problema può essere risolto con una gestione implicita della memoria, ad esempio mediante un memory pool.
- **Compatibilità con più piattaforme:** la filosofia alla base di OpenFOAM mira a garantirne l'accessibilità al più ampio numero possibile di utenti. In passato, ciò è stato ottenuto utilizzando lo standard C++; tuttavia per le GPU non è ancora emerso un approccio vendor-agnostic equivalente.

Sulla base di queste considerazioni, CINECA ha avviato lo sviluppo di SPUMA, con l'obiettivo di realizzare un porting completo di OpenFOAM su GPU.

L'approccio scelto da CINECA si ispira al recente *porting* su GPU del codice NEKO[2], una moderna implementazione del noto codice CFD ad alto ordine Nek5000: diversi *backend* possono essere aggiunti al framework per poterlo eseguire su vari tipi di hardware (ad esempio, NVIDIA, AMD, INTEL), mentre un unico *frontend* viene utilizzato in tutto il codice. Il *frontend* non ha dipendenze da librerie di terze parti, che sono confinate nei vari *backend*.

Questo livello di astrazione è stato combinato con l'adozione della unified memory, una funzionalità hardware/software disponibile sui più recenti acceleratori GPU che consente di allocare uno spazio di indirizzamento della memoria accessibile sia dalla CPU sia dalla GPU. Il risultato è un'implementazione minimamente invasiva e vendor-agnostic. La soluzione proposta può essere eseguita sia su GPU discrete sia sulle più recenti APU, che implementano la unified memory a un livello ancora più profondo. Allo

[2] N. Jansson, M. Karp, A. Podobas, S. Markidis, P. Schlatter, Neko: A modern, portable, and scalable framework for high-fidelity computational fluid dynamics (Computers & Fluids, 2024) https://doi.org/10.1016/j.compfluid.2024.106243

stesso tempo, questo approccio può essere adattato a qualsiasi paradigma GPU disponibile (CUDA, HIP, OpenMP, OpenCL, ecc.).

Ulteriori dettagli sulla metodologia utilizzata per lo sviluppo del codice SPUMA sono disponibili nella recente pubblicazione degli autori.[3]

[3] S. Bnà, G. Giaquinto, E. Fadiga, T. Zanelli, F. Bottau, SPUMA: A minimally invasive approach to the GPU porting of OPENFOAM® (Computer Physics Communications, 2026) https://doi.org/10.1016/j.cpc.2025.110009

# Migliorare le simulazioni CFD con SPUMA: simulazioni più grandi che richiedono meno risorse hardware

Una delle principali sfide nel calcolo parallelo è ottenere algoritmi a scalabilità “quasi perfetta”, ossia algoritmi che mantengano l'efficienza al variare del numero dei processi. Per problemi di grandi dimensioni questo può essere difficile a causa dell’overhead di comunicazione. In OpenFOAM il problema è aggravato dalla lentezza nel processo seriale di decomposizione della mesh, un’operazione che può richiedere diversi giorni o risultare addirittura impossibile.

Problemi complessi, come le simulazioni ad alta fedeltà, richiedono un elevato numero di celle. In questi casi, può essere necessario scomporre il dominio di calcolo in decine di migliaia di partizioni, ognuna delle quali viene assegnata ad un diverso core CPU. Come già accennato, l'aumento del numero di processi oltre una certa soglia diminuisce l'efficienza parallela di un codice come OpenFOAM, e aumentare ulteriormente le risorse computazionali porta sempre meno vantaggi. Inoltre, le operazioni di input/output in OpenFOAM sono segregate, il che significa che ogni processo scrive su un insieme diverso di file. Per le simulazioni che girano su migliaia di core, questo comporta un carico notevole sul file system.

Laddove le CPU funzionano meglio su porzioni più piccole del dominio di calcolo, le GPU, grazie all'elevata larghezza di banda della memoria e all'elevato numero di unità di calcolo, funzionano meglio quando ricevono grandi quantità di dati. Mentre per le CPU il numero ottimale di celle è dell'ordine delle migliaia, per le GPU è dell'ordine dei milioni. Ciò significa che, per le simulazioni di grandi dimensioni e ad alta fedeltà eseguite su GPU, il numero di partizioni necessarie per decomporre il dominio di calcolo è notevolmente ridotto, alleviando i problemi legati alla scalabilità parallela menzionati in precedenza. In questo senso, una versione di OpenFOAM pronta per le GPU estende la gamma di applicazioni del software a problemi

molto più grandi, che sarebbero stati impraticabili o difficoltosi da gestire con le CPU.

Finora, le simulazioni ad alta fedeltà su larga scala richiedono strumenti di simulazione specializzati di alto ordine e ad alte prestazioni, sviluppati per lo più in ambito accademico e limitati nella loro gamma di applicabilità a geometrie semplici e casi specifici. Spesso, queste limitazioni li rendono inadatti alle applicazioni industriali. La capacità, resa possibile da questo porting su GPU, di utilizzare uno strumento versatile e di uso generale come OpenFOAM per simulazioni su larga scala le rende alla portata di una base di utenti molto più ampia. Utenti industriali più piccoli, che non sarebbero stati in grado di eseguire simulazioni di questa portata, possono ora includerle nei loro flussi di lavoro. Ciò crea vaste opportunità di collaborazione tra l'ecosistema delle PMI europee e l'infrastruttura HPC del continente.

# Il vantaggio Agile: ridisegnare i flussi di lavoro di interazione umana per un codice CFD più rapido e di alta qualità

Durante lo sviluppo di SPUMA, la struttura organizzativa è stata modificata per aderire ai principi Agile, offrendo al team un approccio più strutturato alla gestione delle attività.

Nell'Agile SCRUM, un flusso di lavoro iterativo progettato per fornire miglioramenti incrementali garantendo al tempo stesso un'elevata adattabilità, il lavoro viene organizzato in brevi sprint che consentono ai team di ridefinire continuamente le priorità in linea con l'evoluzione delle tecnologie e delle esigenze degli stakeholder, integrare i feedback e riconsiderare gli obiettivi man mano che il progetto avanza. Questo framework aumenta la trasparenza, supporta decisioni informate e favorisce una comunicazione regolare. Dopo 14 sprint, questo approccio ha dimostrato che la ridefinizione dei flussi di interazione tra le persone è un fattore chiave per migliorare i processi di sviluppo, rendendoli nel complesso più rapidi ed efficienti.

**La metodologia Agile**

Ogni sprint inizia con una riunione in cui il team ne definisce gli obiettivi e organizza le attività, aggiornando al contempo il product backlog, l'elenco prioritizzato di tutte le funzionalità da implementare e i task previsti per il progetto. Ogni giorno si tiene una riunione breve di 30 minuti a cui partecipano lo Scrum Master (SM) e gli sviluppatori CFD. Lo SM, una figura non tecnica, fa in modo che le discussioni restino focalizzate sugli obiettivi e nei tempi fissati.

Ogni sprint dura una settimana e si conclude con una riunione di un'ora, durante la quale il team presenta i task completati, discute le criticità incontrate e condivide le soluzioni adottate. Questo incontro funge sia da *revisione* dello sprint sia da *retrospettiva*, offrendo una visione chiara dei progressi e aiutando a identificare possibili miglioramenti per il ciclo di lavori

successivo. Il processo termina con un ritorno al backlog, dal quale vengono selezionati i nuovi task.

Nel nostro caso, le responsabilità del Product Owner (PO) sono affidate al Coordinatore del Team di Sviluppo. Il PO supervisiona le priorità e garantisce coerenza tra lo sviluppo e gli obiettivi più ampi del progetto. Un Project Manager senior svolge il ruolo di Revisore, integrando elementi provenienti dai progetti europei che utilizzano SPUMA e allineandoli alle esigenze degli stakeholder.

**Un approccio innovativo e non convenzionale**

L'adozione dell'Agile è stata motivata dalla necessità di aumentare l'efficienza e sostenere un processo di sviluppo più strutturato e trasparente. Oltre al suo carattere iterativo, l'Agile è stato introdotto per fornire un'organizzazione più chiara, aumentare la responsabilizzazione individuale e incoraggiare un atteggiamento partecipativo, mantenendo al contempo una forte dinamica collaborativa. Queste pratiche supportano anche la gestione dei conflitti, favorendo una comunicazione aperta e offrendo occasioni regolari di revisione esterna, non esclusivamente tecnica. Nel nostro contesto, adottare un approccio Agile rappresenta una scelta non convenzionale.

Il team SPUMA è composto principalmente da professionisti scientifici più che da sviluppatori software a tempo pieno: oltre allo sviluppo del codice, i membri del team partecipano a diversi progetti di ricerca europei, tengono corsi di formazione e applicano SPUMA ad attività scientifiche. Introdurre Agile in un ambiente di questo tipo rappresenta un importante passaggio verso un modo di lavorare più adattivo, collaborativo e iterativo, che permette al team di bilanciare lo sviluppo software con impegni scientifici, mantenendo un allineamento costante tra attività eterogenee.

Un elemento importante di innovazione in questo flusso di lavoro è l'estensione dell'approccio Agile oltre lo sviluppo interno di SPUMA ai progetti di ricerca europei in cui il software è utilizzato. Questi progetti si sono tradizionalmente basati su strutture di tipo waterfall, "a cascata",

caratterizzate da fasi lineari e scarsa flessibilità. Introdurre Agile in questo contesto rappresenta un cambiamento significativo verso un modello più adattivo e collaborativo.

Adottare il framework SCRUM permette di:

1. Minimizzare i rischi legati alla scoperta tardiva di problemi o alla non conformità con gli obiettivi.
2. Mantenere flessibilità in un mercato tecnologico in rapida evoluzione.
3. Adeguarsi a priorità industriali e di ricerca in continuo cambiamento.
4. Fornire soluzioni che soddisfino obiettivi progettuali ed esigenze degli stakeholder.

**Prima e dopo**

Prima dell'adozione dell'Agile, il team seguiva una struttura quasi piramidale basata sulla seniority, in cui le attività erano principalmente definite da un individuo chiave, anziché da un processo chiaro che valorizzasse le priorità degli stakeholder. L'interazione del team avveniva soprattutto durante riunioni mensili plenarie, mentre le attività quotidiane erano per lo più indipendenti. Le attività erano prevalentemente orientate alla ricerca e lo sviluppo veniva svolto principalmente dai membri senior, con una collaborazione limitata. Il *GitLab issue tracker*, pur essendo pensato per la segnalazione di bug software, era stato riadattato a sistema generale di gestione dei task per monitorare lo sviluppo e assegnare responsabilità.

Con l'adozione dell'Agile, la struttura e il flusso di lavoro del team si sono trasformati in un'organizzazione completamente orizzontale, in cui tutti i membri possono proporre nuovi task. La responsabilità del coinvolgimento degli stakeholder è condivisa: i membri impegnati in progetti esterni collaborativi comunicano direttamente con i partner di progetto. Sono stati introdotti nuovi ruoli, come lo Scrum Master e il Reviewer, per garantire supervisione e allineamento. Le interazioni del team sono diventate più strutturate grazie a riunioni regolari come gli sprint planning, i daily stand-up e le sprint review, che scandiscono il ritmo del lavoro, creano

opportunità di feedback reciproco e favoriscono una più rapida convergenza sulle soluzioni. Di conseguenza, i membri del team hanno sviluppato una maggiore consapevolezza delle attività degli altri e una comprensione più chiara degli obiettivi comuni. L'adozione di un Planner non strettamente tecnico ha migliorato il monitoraggio dello sviluppo e la coordinazione, definendo le attività e gestendone avanzamento e priorità.

Un poster[4] dedicato all'adozione della metodologia Agile per SPUMA è stato presentato al terzo OpenFOAM Italian Workshop nel 2026.

[4] V. Sibilia, E. Pascolo, S. Bnà, T. Zanelli, G. Giaquinto, A. Piccolo, C. D. Arlandini, *The Agile Advantage for the SPUMA code Reshaping Human Interaction Workflows for Faster and High-Quality CFD Software Development* (2026)

# Showcase del potenziale di SPUMA: caso d'uso industriale nel settore automotive

### Scalabilità parallela

I risultati di *strong scaling* sul sistema HPC Leonardo (vedi Tabella 1) mostrano che SPUMA è significativamente più veloce rispetto a OpenFOAM standard (v2412).

**Table 1: Specifiche hardware del sistema HPC Leonardo**

| Partizione | **Booster** | **DCGP** |
|---|---|---|
| Modello | Atos BullSequana X2135 "Da Vinci" single-node GPU blade | Atos BullSequana X2140 three-node CPU blade |
| Numero di rack | 116 | 22 |
| Nodi | 3456 | 1536 |
| Processori | Single socket 32 cores Intel Ice Lake CPU.<br>1 x Intel Xeon Platinum 8358, 2.60GHz TDP 250W | Dual socket 56 cores Intel Sapphire Rapids CPU.<br>2 x Intel Xeon Platinum 8480p, 2.00 GHz TDP 350W |
| Acceleratori | 4 x NVIDIA Ampere GPU/nodo, 64GB HBM2e NVLink 3.0 (200GB/s) | - |
| Numero di CPU core | 32 core/nodo | 112 core/nodo |
| RAM | 512 (8x64) GB DDR4 3200 MHz | 512 (16 x 32) GB DDR5 4800 MHz |

Il caso test utilizzato per queste simulazioni è il DrivAer nella configurazione di raffreddamento aperto-chiuso, disponibile nella repository HPC di OpenFOAM. Rappresenta una simulazione RANS automobilistica di aerodinamica esterna, composta da 236 milioni di celle (vedi Figura 1). La Figura 3 mostra le linee di corrente del flusso attorno alla geometria. Questo test dimostra come SPUMA possa interfacciarsi efficacemente con librerie di algebra lineare di terze parti, in questo caso AMGX (sviluppata e rilasciata

da NVIDIA), senza il costo di ulteriori copie host-device, per trasferire la soluzione dell'equazione di pressione al suo framework multigriglia algebrico.

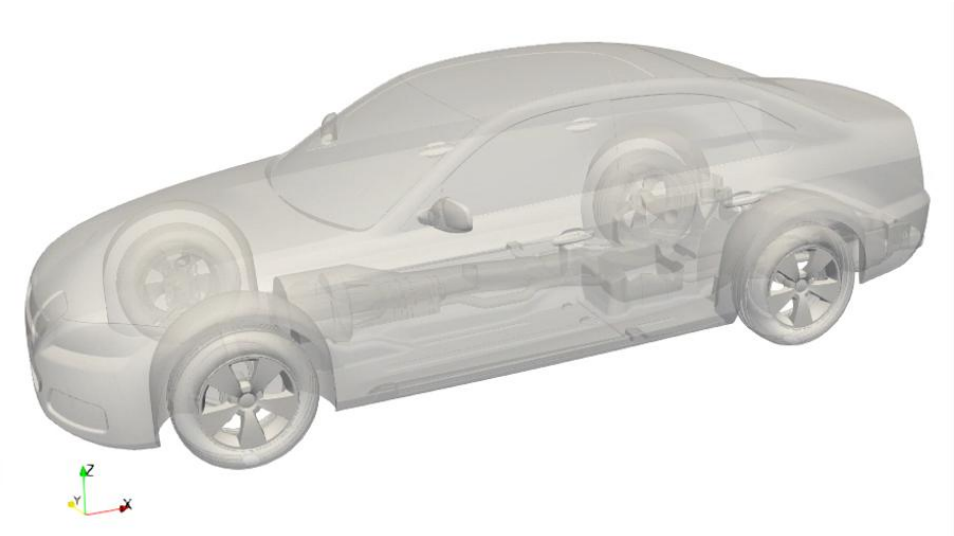


**Figura 1: Geometria DrivAer, immagine cortesia di Charles Mockett, Hendrik Hetmann e Felix Kramer (Upstream CFD GmbH), 2022-2023**

Dai risultati riportati nella Figura 2 emerge che una singola GPU A100 di Leonardo è equivalente a circa 220 core CPU della partizione Data Centric General Purpose di Loenardo (vedi Tabella 2). Pertanto, 8 GPU sono più veloci di 1600 core.

### Test del solutore multigriglia

Il caso di test DrivAer open-closed, composto da 236 milioni di celle, è stato utilizzato anche per valutare le prestazioni del solutore lineare multigriglia geometrico/algebrico (GAMG) nativo di OpenFOAM su GPU. A differenza di un miltigriglia puramente algebrico come AMGX, questo solutore utilizza informazioni geometriche della mesh per costruire una gerarchia di matrici via via più grossolane sulle quali viene eseguito lo smoothing della soluzione. Sulle GPU questo comporta sfide specifiche, poiché le mesh più grossolane sono troppo piccole per sfruttare in modo efficace le risorse disponibili. Possibili soluzioni includono l'agglomerazione delle partizioni di dominio oppure il trasferimento dell'elaborazione delle mesh più grossolane alla

CPU; tuttavia, questi sviluppi sono stati rinviati a lavori futuri. D'altro canto, sono stati investiti sforzi nell'individuare i parametri ottimali per l'esecuzione del solutore GAMG su GPU.

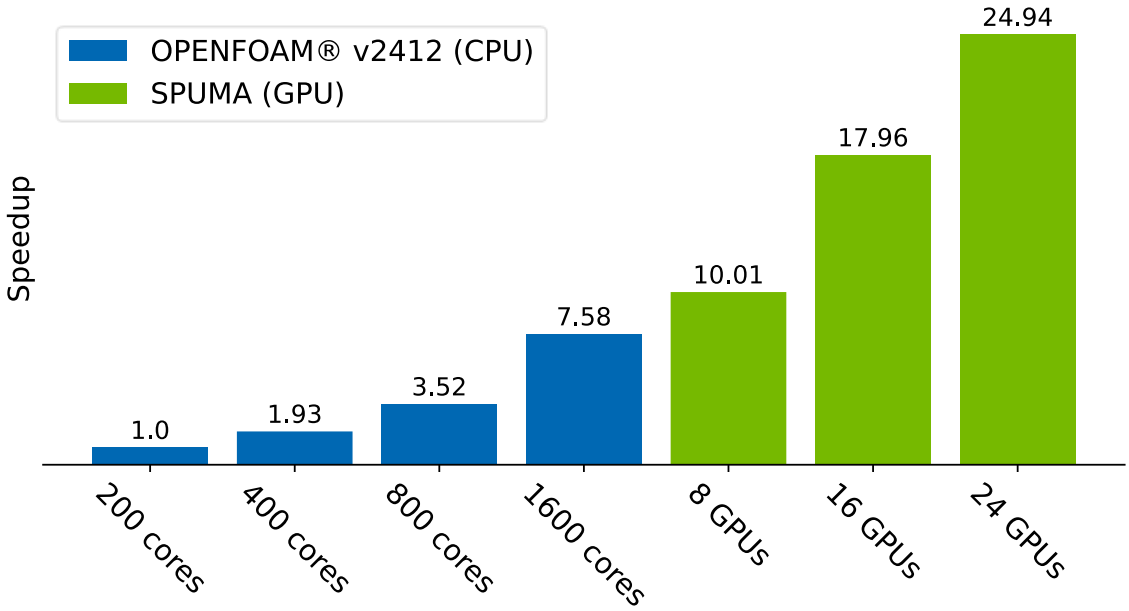



**Figura 2: Risultati dello strong scaling eseguito sul caso HPC DrivAer: in ordinata lo speedup relativo alla simulazione con 200 core CPU**

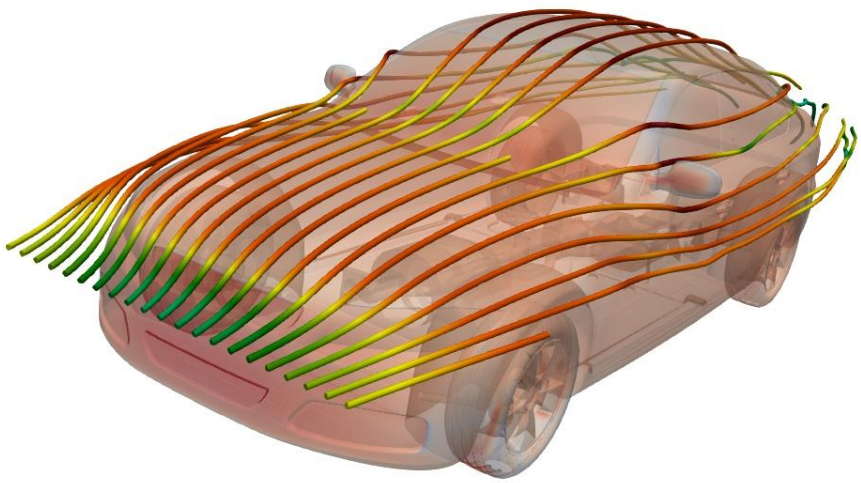


**Figura 3: Visualizzazione delle linee di flusso attorno alla geometria del DrivAer, quest'ultima colorata col campo di pressione**

I test del solutore GAMG sono stati eseguiti sulla partizione Booster del cluster Leonardo (si veda la Tabella 1) e sulla partizione GPU di LUMI (si veda la Tabella 2). La Tabella 3 mostra un confronto tra le diverse configurazioni di esecuzione sui due cluster.

**Table 2: Hardware specifications of LUMI HPC system**

| Partition | **LUMI-G** | **LUMI-C** |
|---|---|---|
| Nodes | 2978 | 2048 |
| Processors | single socket 64 cores AMD Trento CPU.<br>1 x AMD EPYC 7A53, 3.5GHz | Dual socket 64 cores AMD Milan CPU.<br>2 x AMD EPYC 7763, 2.45 GHz base, 3.5 GHz boost |
| Accelerators | 4 x AMD MI250X GPUs 2nd Gen AMD CDNA architecture 2 x 64GB HBM2e (3200 GB/s) | - |
| Cores | 64 cores/node | 128 cores/node |
| RAM | 512 (8x64) GB DDR4 | 256 (8 x 32) GB (or higher) |

**Table 3: Total simulation times in seconds for GAMG tests on Leonardo and LUMI**

| **Cluster partition** | **Floating point precision** | **Velocity solution** | **2 Nodes (8 GPUs)** | **4 Nodes (16 GPUs)** |
|---|---|---|---|---|
| Leonardo Booster | Double (DP) | Segregated | 7951,89 | 4780,7 |
| Leonardo Booster | Double (DP) | Coupled | 7532,39 | 4475,62 |
| Leonardo Booster | Mixed (SPDP) | Segregated | 6091,25 | 3701,34 |
| Leonardo Booster | Mixed (SPDP) | Coupled | 5086,79 | **3413,30** |
| LUMI-G | Double (DP) | Segregated | 8343,20 | 5276,77 |
| LUMI-G | Double (DP) | Coupled | 8919,76 | 5486,10 |
| LUMI-G | Mixed (SPDP) | Segregated | 6608,27 | 4471,05 |
| LUMI-G | Mixed (SPDP) | Coupled | 5697,13 | 3854,75 |

Su LUMI, ogni GPU è composta da due Graphics Compute Dies, che vengono riconosciuti dal *job scheduler* come dispositivi separati. Di conseguenza, mentre su Leonardo il dominio viene partizionato in 8 e 16 parti rispettivamente per le esecuzioni su 2 e 4 nodi, su LUMI il dominio è suddiviso in 16 e 32 parti, influenzando la scalabilità.

I test confrontano due strategie di soluzione per la velocità: quella segregata, come nello standard OpenFOAM, e quella accoppiata, in cui tutte le componenti vengono risolte insieme, evitando l'overhead legato alle soluzioni separate. La Tabella 3 mostra i vantaggi dell'utilizzo di un approccio a precisione mista: precisione singola per l'assemblaggio delle matrici e doppia precisione per la soluzione. Questo aspetto è particolarmente rilevante nel contesto del calcolo su GPU, poiché le tendenze attuali dell'hardware mostrano una preferenza per l'ottimizzazione delle operazioni in virgola mobile a precisione più bassa.

# Il potenziale di SPUMA per ridurre tempi e costi

I vantaggi offerti da SPUMA non si limitano a una maggiore accessibilità a simulazioni di grandi dimensioni e ad alta fedeltà. Per problemi di pari dimensione, l'hardware GPU fornisce un'accelerazione significativa rispetto alle CPU. Ciò è particolarmente interessante per le applicazioni che richiedono tempi di risposta rapidi, come le previsioni meteorologiche o il settore delle corse automobilistiche. Inoltre, va sottolineato che le GPU hanno un costo energetico per FLOPS decisamente inferiore rispetto alle CPU tradizionali, riducendo sia l'onere economico che l'impatto ambientale delle simulazioni CFD.

Inoltre, per tipici casi d'uso industriali, dove le dimensioni delle mesh sono generalmente intorno al milione di celle, una singola GPU può essere sufficiente per una vasta gamma di applicazioni. Nel prossimo futuro, una workstation con una o due GPU di fascia media installate potrebbe risultare un'alternativa economica a un server dedicato basato su CPU per i flussi di lavoro di simulazione di molte piccole e medie imprese.

L'adozione della metodologia Agile ha migliorato lo sviluppo di SPUMA incrementando la trasparenza all'interno del team, aumentando la coordinazione e permettendo risposte più rapide all'evoluzione dei requisiti. Il passaggio da un modello basato sulla seniority ad una struttura collaborativa ha promosso una migliore comunicazione e maggiore responsabilizzazione dei membri del gruppo. La metodologia Agile, pertanto, ha reso il processo di sviluppo più efficiente ed adattabile, dimostrando la sua utilità anche per i team, spesso di dimensioni ridotte, che si occupano di software scientifico.