Reunión seguimiento
15/06/2022

ArTeCS

UNIVERSIDAD COMPLUTENSE MADRID

Comunidad de Madrid

UNIÓN EUROPEA
Fondos Estructurales
Invertimos en su futuro
UNIÓN EUROPEA
Fondo Social Europeo
El Fondo Social Europeo invierte en tu futuro

Obj 2: Explotación paralelismo nodo y aceleradores

- Modelos de Programación
  - *Paralelismo de tareas*
  - *Patrones - GrPPI (UC3M)*
- Software de Sistema
  - *Planificación y Gestión de Recursos en Multicore*
  - *Maleabilidad*
  - **Coplanificación (UCM)**
- Aceleradores
  - **Aceleradores Aritméticos (UCM)**
  - *Aceleradores RISCV*
  - **Explotación y Portabilidad (UCM)**

Obj 2: Explotación paralelismo nodo y aceleradores

- Compromisos 18 meses
  - E2.1 Herramienta de caracterización de tareas.
  - E2.2 Definición de modelo de programación para explotación de patrones de paralelismo
  - Artículos científicos.
- Compromisos 48 meses
  - E2.3 Metodologías de síntesis para aceleradores tipo FPGA.
  - E2.4 Diseño de Lenguaje Específico para problemas intensivos de datos.
  - E2.5 Evaluación de técnicas predictivas para gestión recursos.
  - H3 Integración en el framework de patrones de paralelismo y soporte para aceleradores
  - H4 Integración en el framework de técnicas predictivas
  - Artículos científicos.
  - 5 tesis doctorales.

Obj 2: Explotación paralelismo nodo y aceleradores

- Resultados
  - Tesis Doctorales
    - Anton Rey, Luis Costero, Adrian Garcia (UCM)
    - Plácido Fernández, Javier Lopez (UC3M)
  - Tesis en curso
    - Daniel Garcia, **Raul Murillo, Youssef El Faqir, Maria José Belda**, David Mallasén (UCM)
  - Publicaciones ~ 15 Journals
  - Entregables (GitHub)
    - GrPPI
    - PMCTrack
    - Deep PeNSieve, PERCIVAL, Flo-Posit

Modelos de Programación basados en Tareas

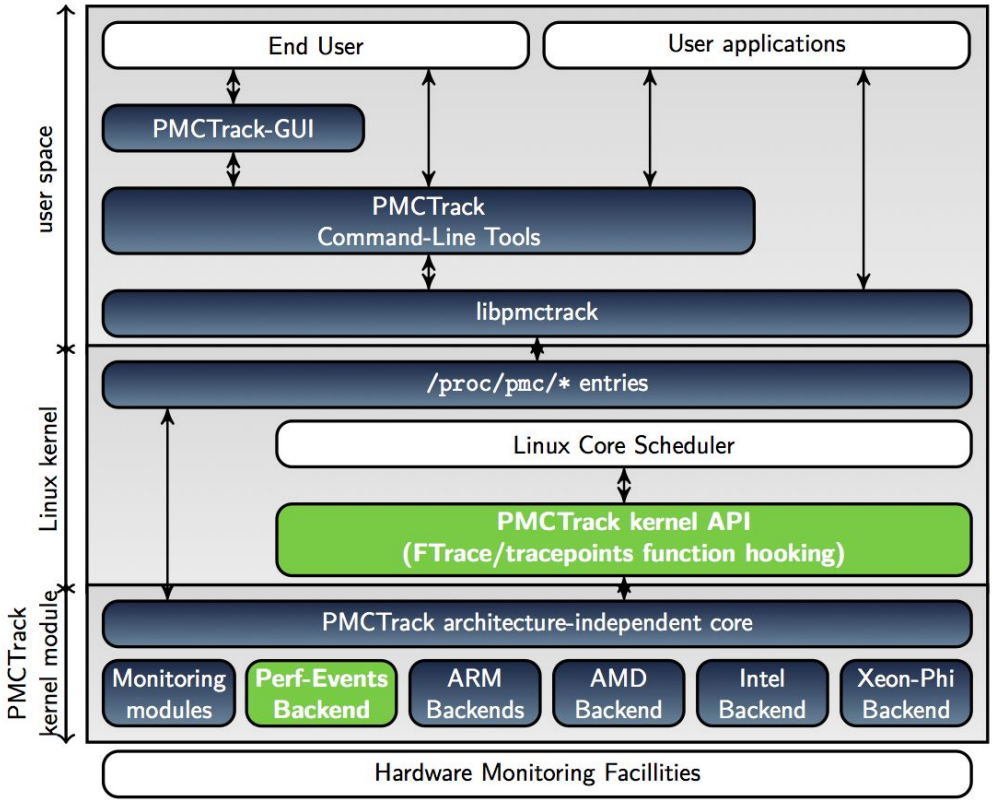
- De código secuencial a paralelismo *Async/ManyTask/Heterogéneo*
  - ***Anotaciones de tareas*** (compilación) para expresar:
    - Equivalencia entre funciones
    - Argumentos de funciones *delegados*
  - Infraestructura de Compilación Basada en *Clang*
  - *Source-to-source: se genera código C++20*
    - *Módulos y Co-rutinas*
  - *Previsión de Beta disponible a finales de año*

```
[[steel::task("gemm")]]
void dense_gemm(A,B,C,n) { ... }

[[steel::task("gemm")]]
void dense_gemm_gpu(A,B,C,n) { ... }

[[steel::task("gemm")]]
void sparse_gemm(A,B,C,n) { ... }

[[steel::task("gemm")]]
[[steel::manage("bs=divisors(n)")]]
void block_gemm(A,B,C,n,bs) { ... }
```

Patrones de Paralelismo. GrPPi

- Work in progress (2021)
  - Nuevos patrones en GrPPi
  - Herramienta de Refactorización
  - Nuevo backend para SYCL
- Transferencia
  - Proyecto con BBVA (NDA)

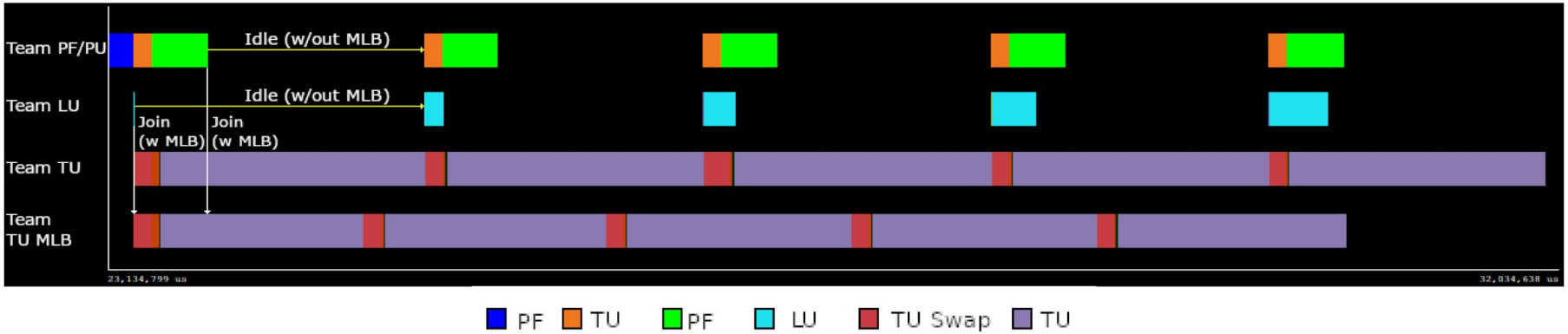
Software de Sistema: PMCTrack v2.0



- Herramienta open-source para Linux
  - https://github.com/jcsaezal/pmctrack
- API para acceso a información de monitorización HW desde SO y espacio de usuario
  - Contadores hardware
  - Cache occupancy y *bandwidth*
  - Consumo energético
  - ...
- Módulo del kernel extensible
- **Nuevas características v2.0:**
  - Ya no precisa parchear el kernel Linux
    - Puede usarse sobre kernels *vanilla* para x86 y arm (32 y 64 bits)
  - Nuevo backend acceso a eventos vía *Perf events*
  - Coexistencia con herramienta *perf*

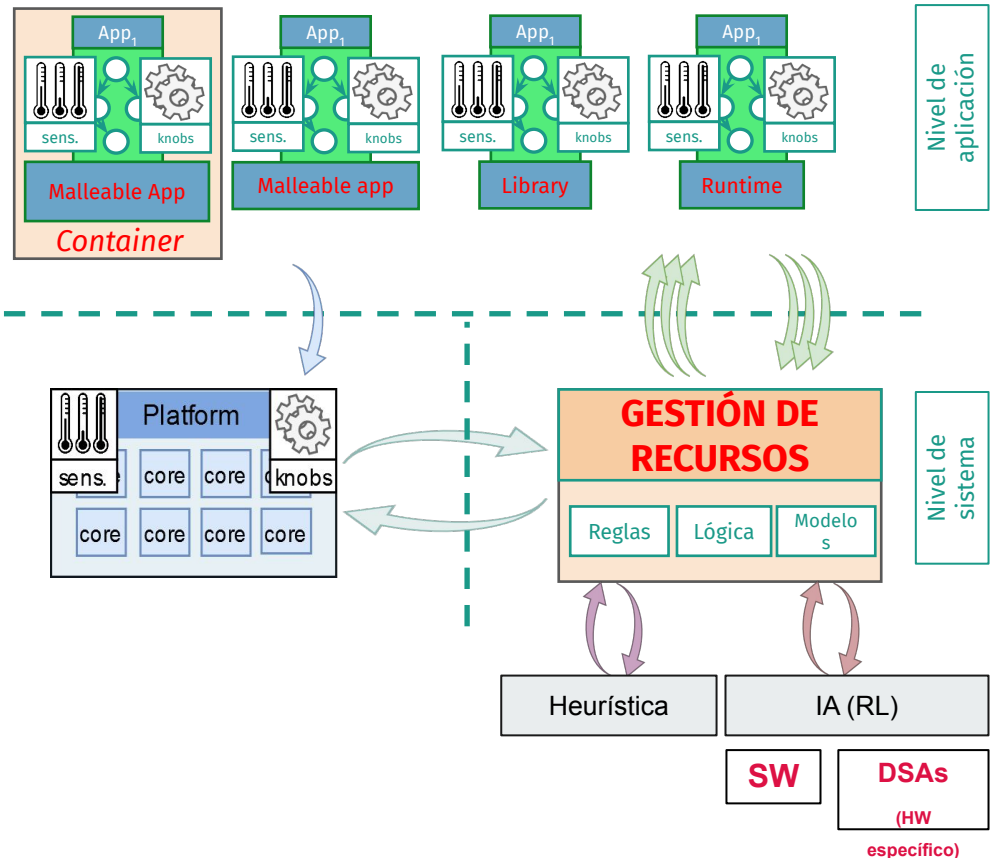
Maleabilidad

- BLAS maleable (BLIS)
  - BLIS (BLAS-Like Instantiation Software):
    - Permite variar el número de threads en un kernel en ejecución (p.e. DGEMM)
- Operaciones maleables
  - Distintas tareas que utilizan BLAS maleable (PF/PU, LU, TU)
  - Se asignan los threads de cada tarea al inicio de la iteración
  - Una vez los recursos son liberados por una tarea se pueden asignar a otra tarea en ejecución
  - Estrategia aplicable a distintas operaciones: QR, LU, INV...



Coplanificación en entornos multi-aplicación

Soporte HW e IA para gestión de recursos en entornos multi-aplicación



Múltiples aplicaciones *maleables*

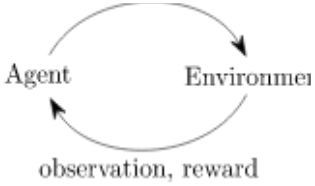
- Modelos de programación basados en **tareas**
  - Exponen *knobs*
  - Varían comportamiento (uso de recursos) bajo demanda
- **Bibliotecas** (e.g. Tensorflow, BLAS) maleables
- **Contenedores - Docker**
  Asignación dinámica de recursos (paralelismo vertical/dentro del contenedor)

Lógica de coplanificación/gestión de recursos

- **Sensoriza** uso de recursos a nivel de sistema/aplicación
- Basada en **heurísticas**
- Basada en **IA** (*Aprendizaje por Refuerzo*)
  - Formulación basada en redes neuronales
  - Entrenamiento/inferencia vía HW
- Línea abierta: Explotación de **aceleradores HW** para toma de decisiones (Google Coral, Intel NCS, IMGTEC NNA, GPUs, ...)

Aprendizaje por Refuerzo. Infraestructura

- Punto de partida:
  - **GYM-AI**: toolkit para desarrollo y comparación de algoritmos de RL. Creación de entornos
  - **Rllib**: framework para entrenamiento de agentes vía Aprendizaje por Refuerzo (RL)
  - Típicamente utilizado para aprendizaje automático en juegos.
  - Función de aproximación basada en redes neuronales (TensorFlow)
- Elementos:
  - Espacio de **acciones**: *inputs* proporcionados al entorno
  - Espacio de **observaciones/estados**: *outputs* proporcionados por el entorno
  - **Recompensa** devuelta por el entorno tras aplicación de acción




```
import gym
env = gym.make("CartPole-v1")
observation = env.reset()
for _ in range(1000):
  env.render()
  action = env.action_space.sample() # Selección de acción.
                                     # Por defecto, acciones aleatoria
                                     # El agente entrenado iría aquí.
  observation, reward, done, info = env.step(action)

  if done:
    observation = env.reset()

env.close()
```

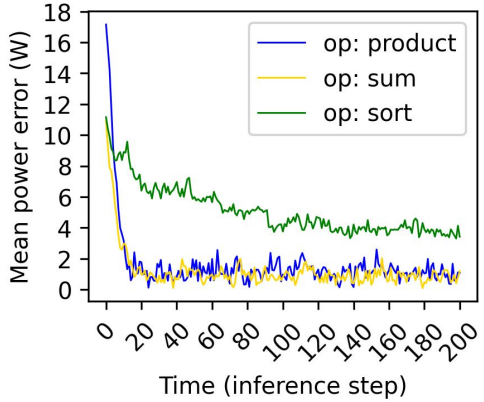
No entrenado (random)

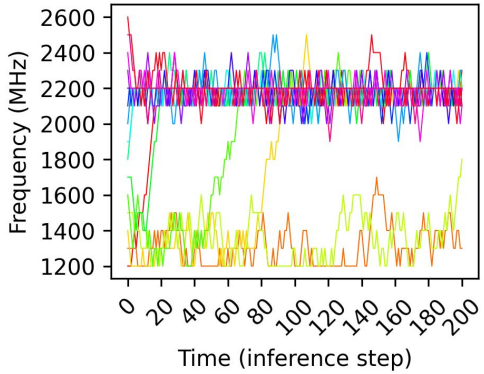
Aprendizaje por refuerzo en la Gestión de Recursos

- Extensión de entornos GYM para adaptarlos a gestión de recursos
  - Manteniendo interfaz
  - Extendiendo interacción con el entorno (e.g. Intel RAPL, HW counters, DVFS, …)
  - Entrenamiento/inferencia vía *RIlib*

- **Ejemplo 1**: *entorno GYM para power capping**
  - Espacio de **acciones**: *modificación de frecuencias (DVFS-cpufreq)*
    - *¿Cómo definir las acciones? Afecta a la calidad y al tiempo de aprendizaje*
  - Espacio de **observaciones/estados**: *monitorización de consumo (e.g. RAPL)*
    - *¿Cómo definir los estados? Afecta a la calidad y al tiempo de aprendizaje*
  - **Función de recompensa**: *"premia" estados destino cercanos a cierto cap*

$$R_{t+1}(s_t, s_{t+1}) = \begin{cases} -1 & \text{if} \quad |s_{t+1} - s_{goal}| > |s_t - s_{goal}| \\ +1 & \text{if} \quad |s_{t+1} - s_{goal}| < |s_t - s_{goal}| \\ +2 & \text{if} \quad s_{t+1} - s_{goal} = 0 \end{cases}$$

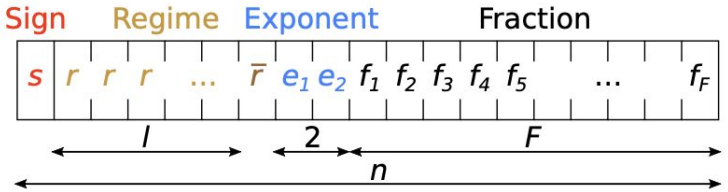


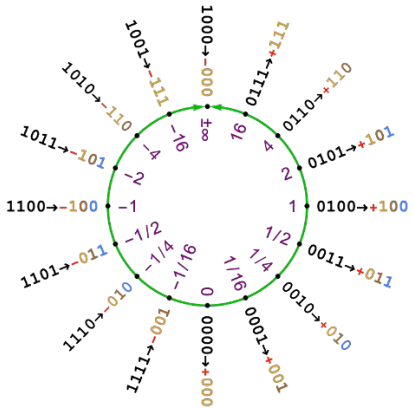


* P. Hernández et al. "Applying Game-Learning Environments to Power Capping Scenarios via Reinforcement Learning". JCC 2022.

Aceleradores aritméticos

- Basados en el formato posit, introducido por John Gustafson en 2017
- Permiten mayor precisión con menor número de bits
  - Disminución de huella de memoria
  - Disminución de área, consumo
- Aplicaciones
  - DNNs
    - **Raul Murillo, Alberto A. Del Barrio, Guillermo Botella: Deep PeNSieve: A deep learning framework based on the posit number system. Digit. Signal Process. 102: 102762 (2020)**
      - https://posithub.org/docs/PDS/PositEffortsSurvey.html
  - Aplicaciones científicas
    - Simulación
    - Aceleradores HW
    - Cores RISC-V custom

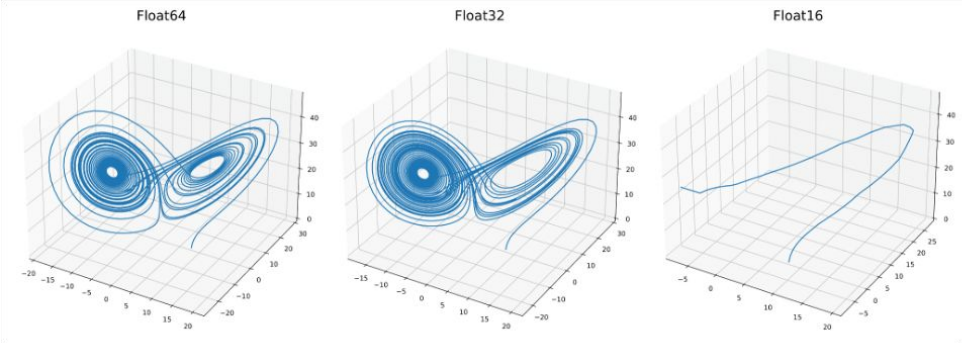




Posits para cálculo científico

Murillo, R., et al. "The Effects of Numerical Precision in Scientific Applications." 2022 Annual Modeling and Simulation Conference (ANNSIM). 2022.

- Estudio de precisión numérica en aplicaciones de HPC.
  - Posits permiten una mayor precisión o usar menos bits.



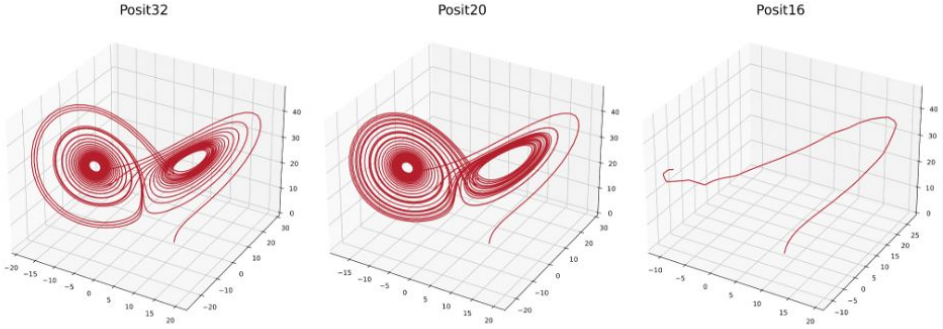
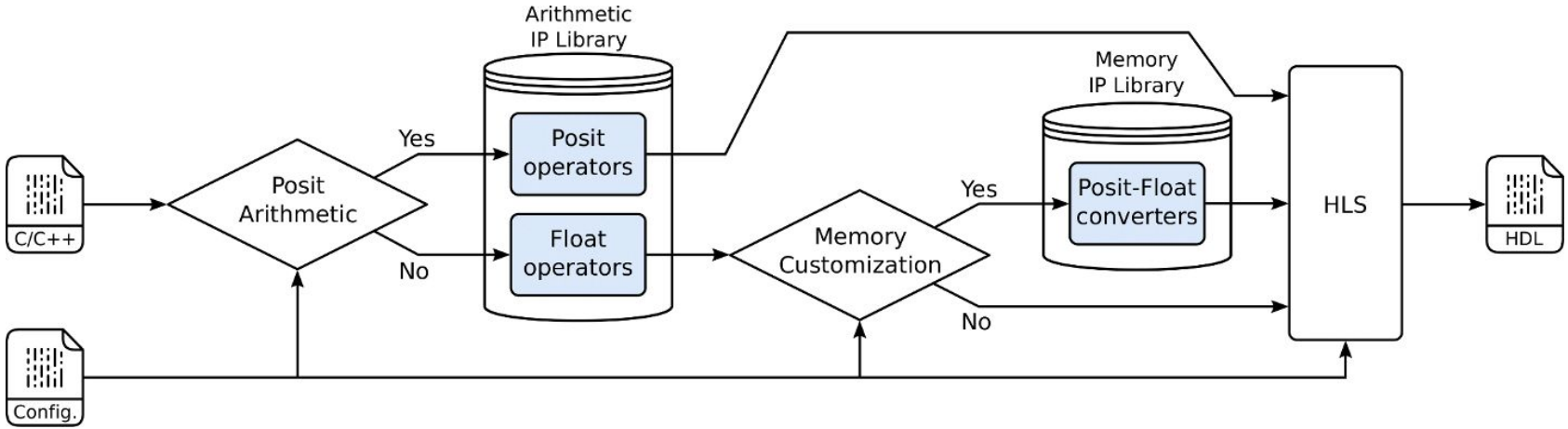


Table 1: Error comparison for different number formats.

Benchmark	Float32	Float16	Bfloat16	Posit32	Posit28	Posit24	Posit20	Posit16	Posit14	Posit12	Posit10	Posit8
Brownian 1D	3.13e-12	1.95e-05	1.03e-03	6.11e-13	4.55e-12	1.74e-10	6.92e-08	8.58e-06	4.30e-05	6.02e-03	1.73e-02	1.45
Brownian 2D	8.77e-12	1.39e-04	2.39e-03	1.01e-12	1.11e-11	3.83e-10	2.11e-07	1.97e-05	1.13e-03	5.39e-03	2.63e-01	1.39
FFT	8.56e-15	7.70e-09	5.55e-07	5.31e-17	1.67e-15	1.52e-13	2.52e-11	5.12e-09	1.01e-07	1.38e-06	2.88e-05	3.49e-04
Lorenz	80.27	63.64	159.39	39.95	72.82	74.01	90.10	76.34	142.03	325.91	319.73	315.35

HLS + Posits

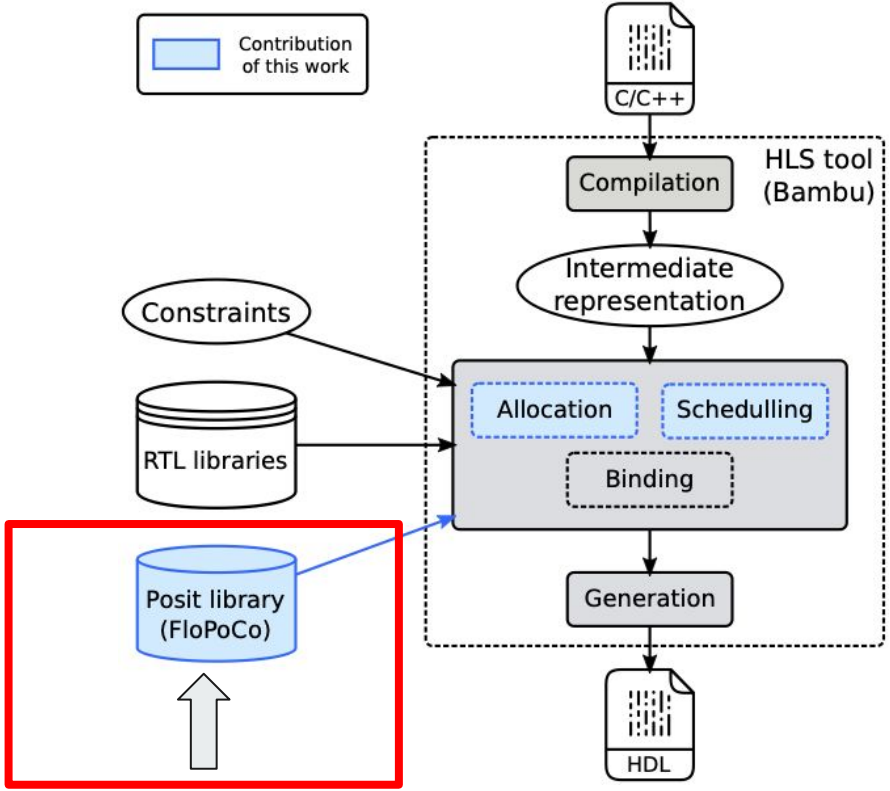
- Generación de aceleradores basados en aritmética posit mediante HLS ⇒ Branch en Bambu (herramienta open-source, https://panda.dei.polimi.it/)
  - Integra con FloPoCo. Trabajo previo
- Permite compatibilidad con flujos convencionales

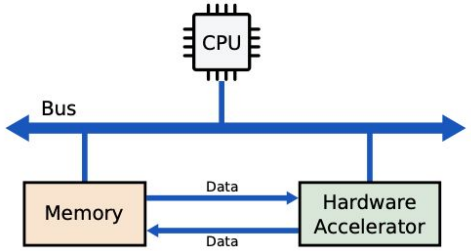


Colaboración con el Politécnico de Milán. Enviado a la conferencia ICCAD 2022 (Core A, Class 2)

HLS + Posits

https://github.com/RaulMurillo/Flo-Posit





Device	Case #1	Case #2	Case #3
Memory	Float	Posit	Posit
Accelerator		Float	

Table 1: Error comparison for different number formats.

Memory precision	Computation precision	Lorenz	Helmholtz
Float32	Float32	239.37	1.20e-11
Float64	Float64	4.63e-08	5.98e-29
Posit32	Float32	239.37	1.20e-11
Posit32	Float64	4.63e-08	5.10e-14
Posit32	Posit32	86.83	1.99e-13
Posit64	Posit64	3.95e-11	1.21e-30

R. Murillo et al. ,"Energy-Efficient MAC Units for Fused Posit Arithmetic," in 2021 IEEE 39th International Conference on Computer Design (ICCD), Oct. 2021, pp. 138–145. doi: 10.1109/ICCD53106.2021.00032.

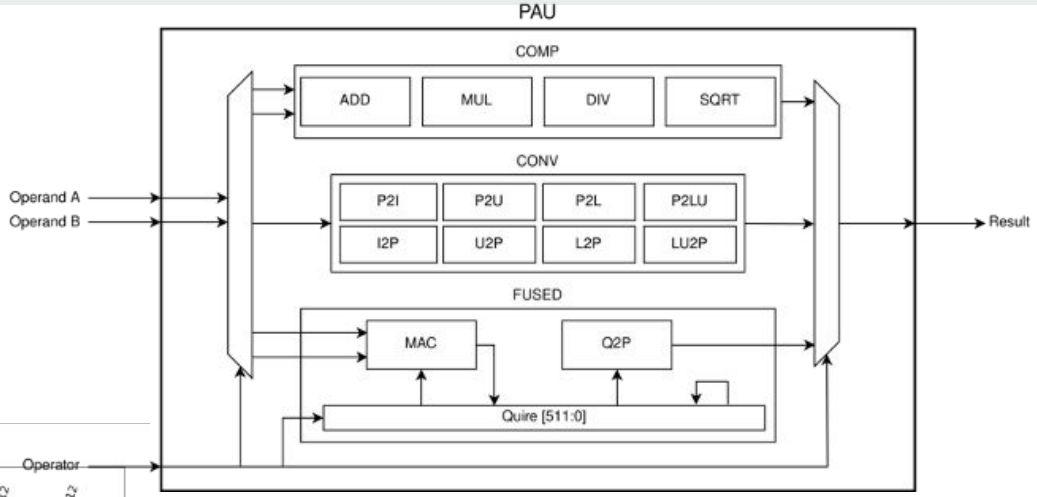
R. Murillo et al., "Comparing Different Decodings for Posit Arithmetic," presented at the Conference on Next Generation Arithmetic (CoNGA), 2022.

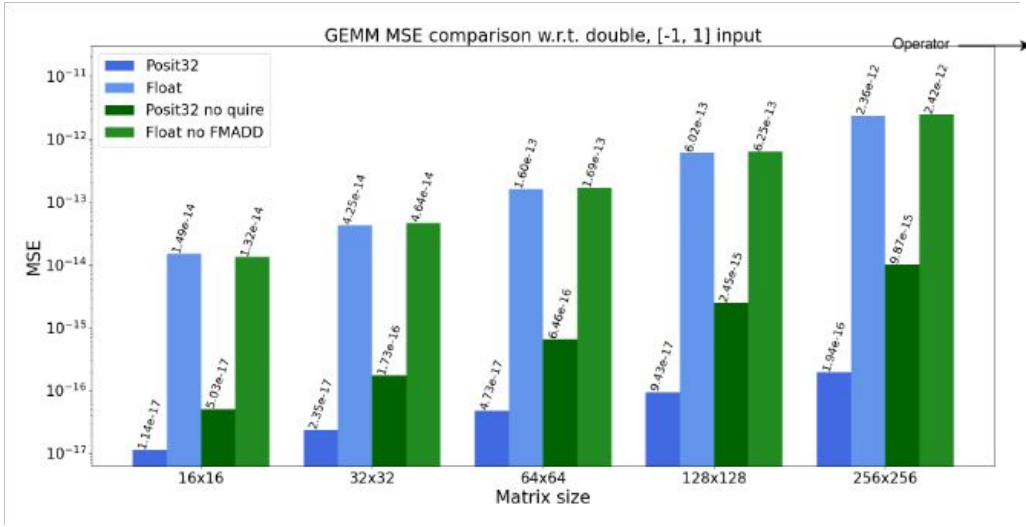
R. Murillo et al., "PLAM: a Posit Logarithm-Approximate Multiplier," in IEEE Transactions on Emerging Topics in Computing, doi: 10.1109/TETC.2021.3109127.

PERCIVAL: Aritmética posit con quire en RISC-V

- Basado en el core CVA6 (Ariane)
- Soporte nativo tanto posit como IEEE 754
- Extensión Xposit para RISC-V en LLVM
- Ejecución a 50MHz en FPGA Kintex-7

https://github.com/artecs-group/PERCIVAL



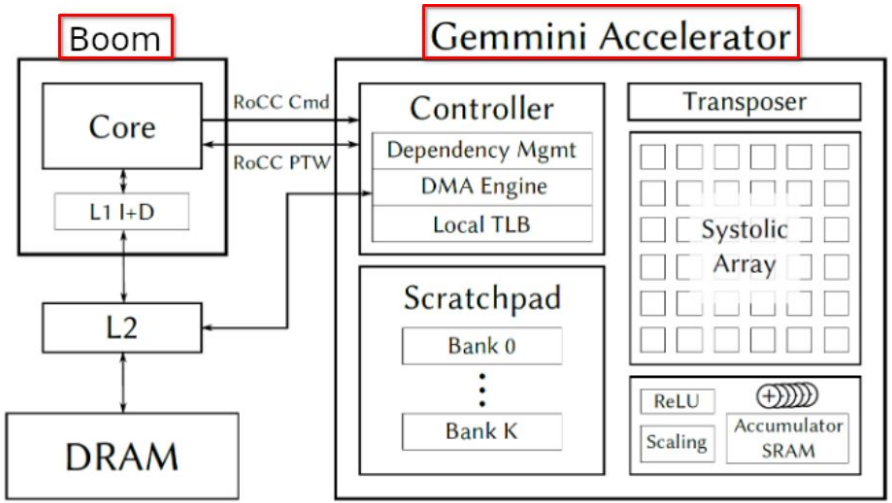


D. Mallasén et al., "PERCIVAL: Open-Source Posit RISC-V Core with Quire Capability". Minor Review en IEEE TETC + ARITH'22. Versión pública en arXiv:2111.15286

Uso de aceleradores específicos de dominio

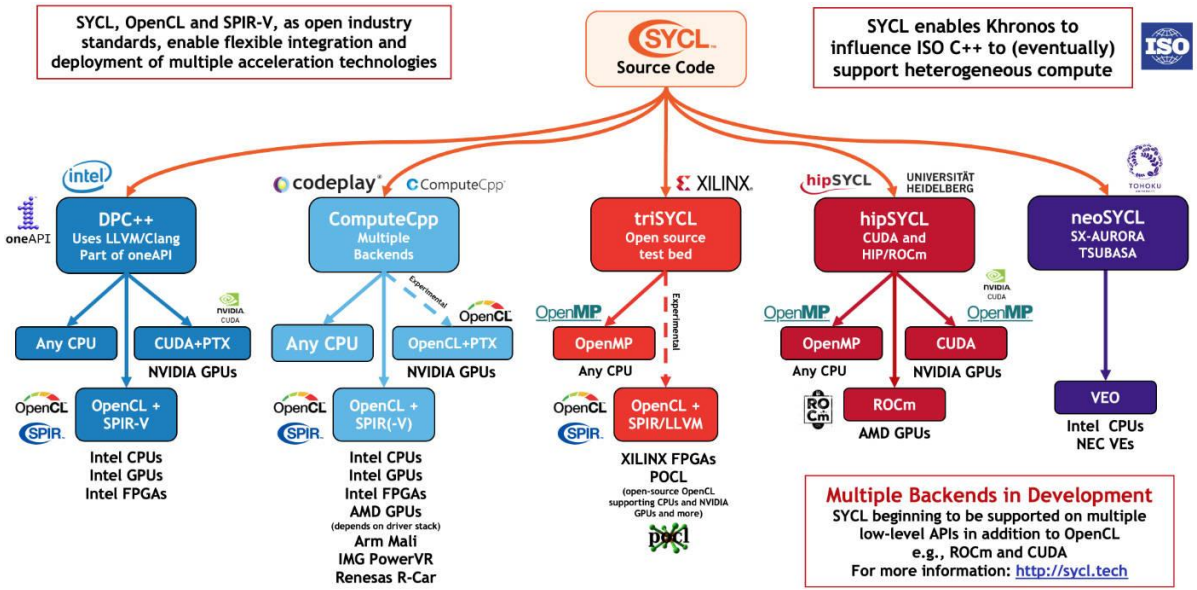
- Objetivo: Acelerar aplicaciones de visión en entornos de coches autónomos utilizando un sistema heterogéneo formado por un core de propósito general y un acelerador específico de dominio

- Implementación: Arquitectura RISC-V sobre FPGAs de Amazon



Explotación y heterogeneidad

- SYCL estándar de programación heterogénea → unificar bajo un mismo modelo de programación el uso de diferentes aceleradores y diferentes fabricantes.

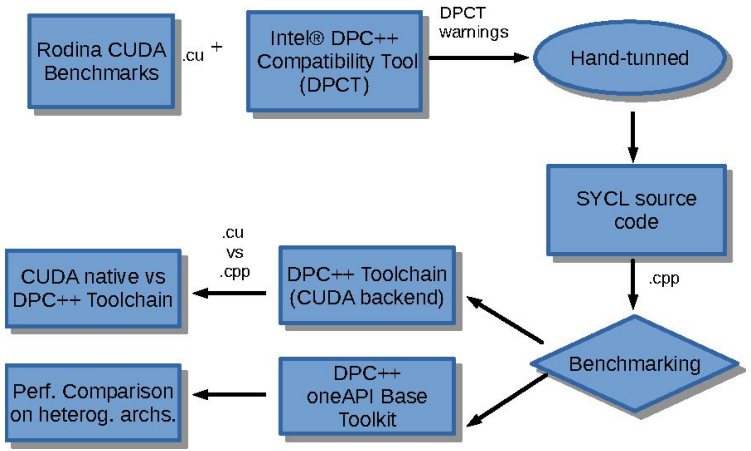


Explotación y heterogeneidad

- SYCL explota el paralelismo de cada arquitectura
- oneAPI es la implementación comercial Intel del estándar SYCL
  - Programación directa en C++: dpcpp
  - Librerías especializadas para diferentes propósitos, ya sea álgebra lineal (oneMKL), analítica de datos (oneDAL), o deep learning (oneDNN)

Evaluación de portabilidad

- Utilización herramienta portabilidad Intel DPCT[1]
  - Data Parallel Compatibility Tool (CUDA→SYCL)
- Evaluación con benchmarks: Rodinia[2]



Applications	Dwarves	Domains	Parallel Model
Leukocyte	Structured Grid	Medical Imaging	CUDA, OMP, OCL
Heart Wall	Structured Grid	Medical Imaging	CUDA, OMP, OCL
MUMmerGPU	Graph Traversal	Bioinformatics	CUDA, OMP
CFD Solver	Unstructured Grid	Fluid Dynamics	CUDA, OMP, OCL
LU Decomposition	Dense Linear Algebra	Linear Algebra	CUDA, OMP, OCL
HotSpot	Structured Grid	Physics Simulation	CUDA, OMP, OCL
Back Propogation	Unstructured Grid	Pattern Recognition	CUDA, OMP, OCL
Needleman-Wunsch	Dynamic Programming	Bioinformatics	CUDA, OMP, OCL
Kmeans	Dense Linear Algebra	Data Mining	CUDA, OMP, OCL
....			

[1] Intel DPCT: https://www.intel.com/content/www/us/en/developer/tools/oneapi/dpc-compatibility-tool.html
[2] GitHub: https://github.com/artecs-group/rodinia-dpct-dpcpp
[3] "Evaluation of Intel's DPC++ Compatibility Tool in Heterogeneous Computing." Journal of Parallel and Distributed Computing 2022, vol 165, pages 120-129.

Evaluación de portabilidad

- Evaluación con benchmarks: Rodinia[2]
  - 20/23 benchmark son traducidos sin gran intervención de programador
  - Operaciones memoria CUDA vs SYCL rendimiento parecido
  - Disparidad de resultados de rendimiento CUDA vs SYCL (nvidia)
    - Algunos overheads de 25%-190%
  - (CPU) OpenMP vs SYCL equivalentes excepto para benchmarks “ligeros” cuyo rendimiento está limitado por syncro (barrier)

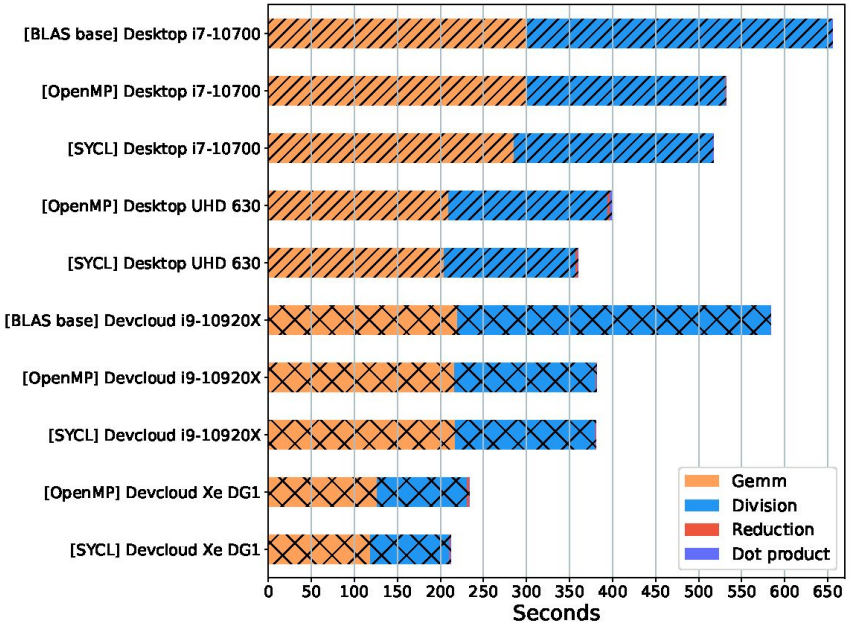
[2] GitHub: https://github.com/artecs-group/rodinia-dpct-dpcpp
[3] "Evaluation of Intel’s DPC++ Compatibility Tool in Heterogeneous Computing." Journal of Parallel and Distributed Computing, volume 165, pages 120-129.

Portabilidad de librerías

- Caso de uso: Algoritmo de factorización matrices NMF
  - Evaluación de diferentes paradigmas y arquitecturas
    - SYCL vs OpenMP
    - CPU vs GPU (integrada/PCIe)
- Implementación
  - Operación más costosa GEMM (MKL)
    - MKL soporta offloading tanto SYCL como OpenMP
  - Kernels ad-hoc: reducción y división punto a punto

Portabilidad de librerías

- CPU (multicores)
  - Rendimiento equivalente SYCL vs OpenMP
  - Líneas de código similar
  - Reducción sencilla
- GPU
  - SYCL (++ perfs)
  - Reducción en árbol (SYCL)
    - Sin adaptación en OpenMP



Evaluación de Portabilidad

- NMF en scikit-learn (no uso de heterogeneidad)
  - Aunque oneAPI dispone de oneDAL no está completa scikit-learn
- Evaluación de otros algoritmos de Machine-Learning como “k-means”
  - Trabajo preliminar en CPU (x86), GPUs (nvidia, intel)
  - Estudio en FPGA mediante oneAPI
  - Integración en librerías como oneDAL