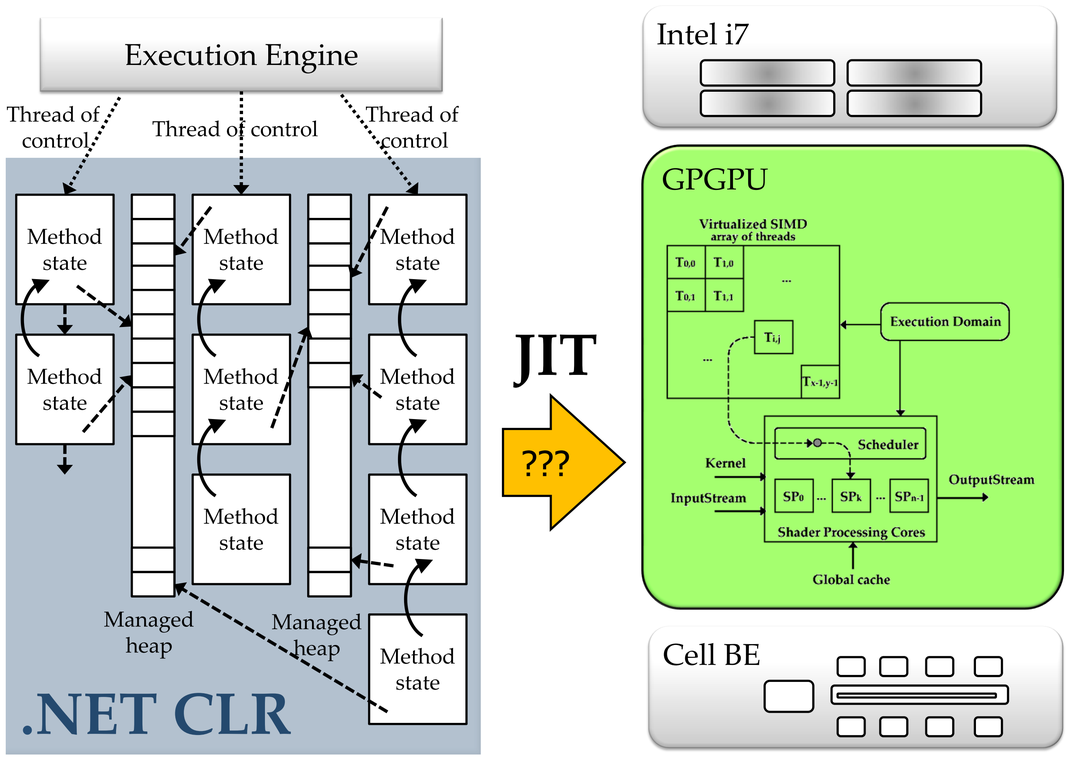
Filling the gap between GPGPUs and Virtual Machine Computational Models

Dittamo Cristian Cisternino Antonio

Computer Science Dept., University of Pisa
Largo B. Pontecorvo 3, 56127, Pisa, Italy
Email: {dittamo, cisterni}@di.unipi.it

MSIL http://cvslab.di.unipi.it/sparc **PTX**

Mapping problem – Limits of the Virtual Machine



Problems:

1. The design of VMs (Sun's JVM, Microsoft's CLR) was influenced by the dominant idea that processors would have maintained a Von-Neumann model while hiding special-purpose aspects.
2. Special-purpose architectures expose quite different parallel computational models and require different programming models.
3. The Just-In-Time compiler cannot target special-purpose architecture features since they are hidden from the abstraction layer provided by the intermediate language (IL).

Mapping: meta-programming + metadata

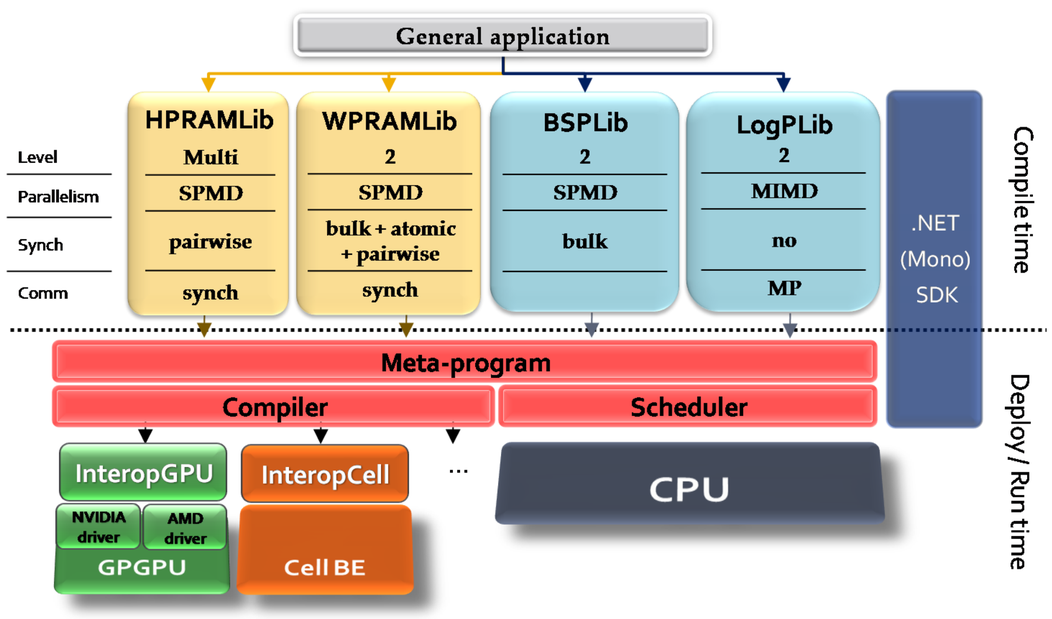
The idea is to define and implement a meta-programming technique that can map the VM stack-based programming model to different models of parallel computation:

1. **without affecting** the general structure of the VM;
2. **by raising** semantic level to eliminate explicit sequencing;
3. **by providing** suitable programming abstractions to define a single and unified programming model without losing expressivity nor forcing the use of a single source language.

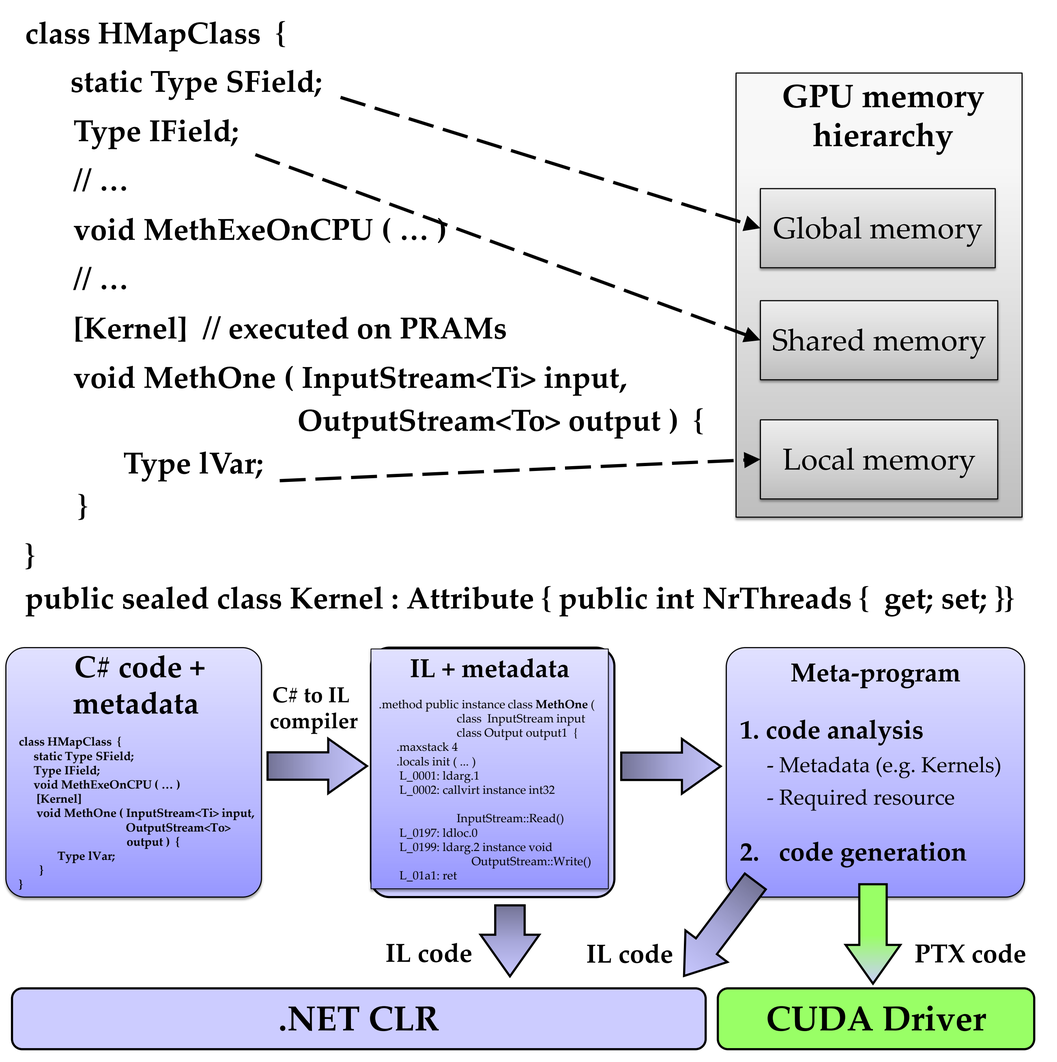
Models of Parallel Computation → **VM computational model**

Our work is not tailored to neither a specific architecture nor a single execution model, but to consider different classes of parallel models:

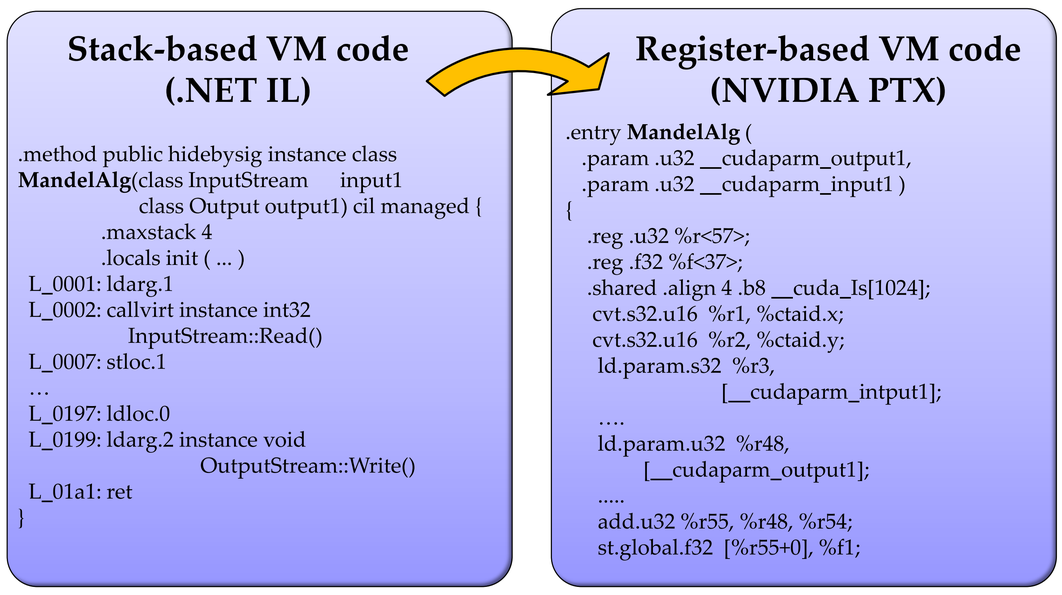
- **Shared memory**, e.g. the Hierarchical PRAM (HPRAM), the Weakly Coherent PRAM (WPRAM);
- **Distribute memory**, e.g. Bulk Synchronous Process (BSP)



NVIDIA GPU-HPRAM model mapping on .NET CLR



Mapping: from Stack to Register VM code



- abstract interpretation of operand stack
- one-one translation rule whenever possible
- stack operations optimization
- branch translation

Conclusion

- ✓ HPRAM and BSP models already mapped on .NET CLR
- ✓ Microsoft IL to NVIDIA PTX code compiler
- ✓ Developing and Debugging at source level such that programmers from the tasks of specifying parallelism, communication and synchronization
- ✓ Approach leverages features of the object-oriented programming (e.g. inheritance) and types
- ✓ Using metadata, programmers can expose to JIT special features of underlying architectures