SciLive

LIVE DOMAIN SPECIFIC PROGRAMMING TOOLS AND LANGUAGES FOR COMPUTATIONAL SCIENCE

Institut de Recherche pour le Développement
FRANCE

Project leader: Serge Stinckwich, IRD, 32 avenue Henri Varagnat, 93140 Bondy, France; serge.stinckwich@ird.fr
5 Partners: UMMISCO, IRD&UPMC, Bondy, France; RMOD, INRIA, Lille, France; G-EAU, IRSTEA, Montpellier, France; CEA/LIST, Paris, France; Yaounde 1 University, Cameroon.

DESCRIPTION

Computational science is rapidly growing and becoming a multidisciplinary field that uses advanced computing capabilities to understand and solve complex problems (in epidemiology, climate change, natural resources management). It is an area of science, which spans many disciplines, but at its core involves the development of models and simulations to understand natural systems. Especially for problems with high socio-ecological interest, mathematical modeling & massively parallel simulations of complex systems can support and safe-guard societies in need. *Yet retrospective analysis on modeling and simulation of environmental threats show us a number of key drawbacks of computational science.* For example in the recent Ebola outbreak in West Africa initial predictions were refuted by a large margin and a constant **need for adaptation and evolution of tools, algorithms and models** during the outbreak period emerged. In such cases of high-performance intensive simulations, a faster cycle of adaptation and evolution of software is nearly impossible, without dedicated infrastructure and tools.

In such a modeling and environmental simulation context, SciLive aims to support non-experts in Computer Science to express their scientific models with the help of domain-specific languages (DSLs) and associated domain specific tools. Inside a dynamic development environment these non-experts will be able to change algorithms or data lively without interrupting their current simulations and without compromising the need of highly efficient computation.

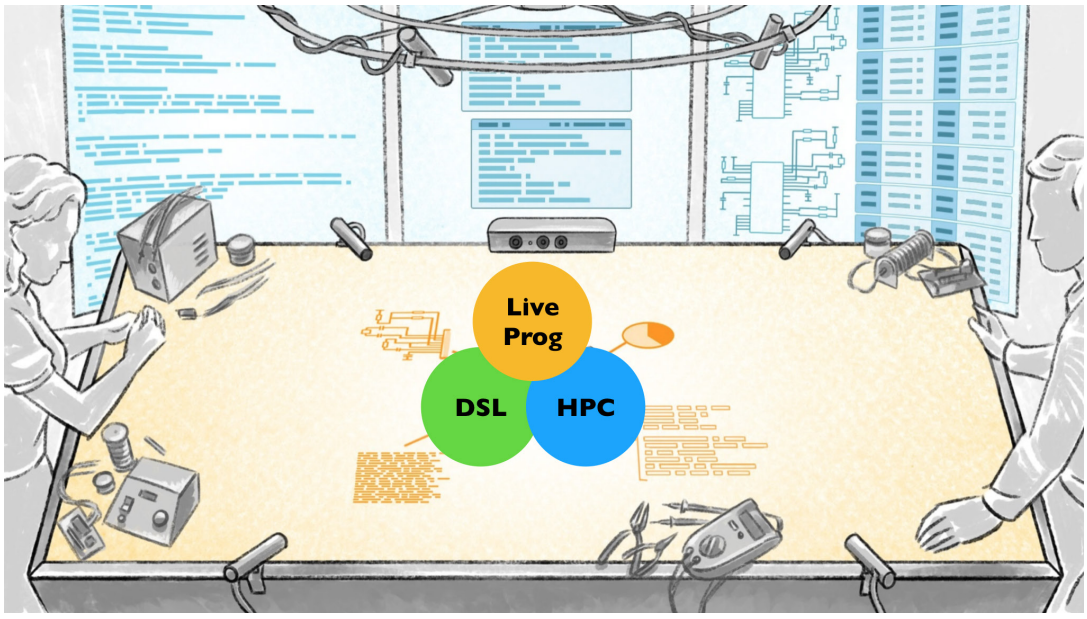

Fig. 1: Combining Domain-Specific Languages, Live Programming and High Performance Computing (extract from Seeing Spaces, Bret Victor)

COMBINING HPC, LIVE PROGRAMMING AND DSL

High-performance computing (HPC) uses supercomputers and parallel processing techniques for solving complex computational problems. Developing simulation software which effectively models scientific phenomena, enable exploration and discovery, and run efficiently on modern **heterogeneous HPC architectures** is a difficult and demanding undertaking. One of the reasons for this is the very slow feedback loop between code development, execution and analysis. **Heterogeneity**, now considered a "must have" for performance and energy efficiency, increases the level of difficulty by increasing the hardware targets (CPUs, GPUs, FPGAs, manycores) and assumes the use of appropriate runtimes (StarPU, OMPSS, xkaapi) and languages (OpenCL, CUDA).

Live programming solutions which have originated with Lisp and Smalltalk systems have recently seen a renewed research and industrial interest due to their educational and productivity potential (see recent Live workshops at ECOOP, ICSE, and SPLASH, live facilities for the Java, Python, and Swift platforms). Live programming is an approach for programming where the traditional edit-compile-run cycle is abandoned in favor of a more fluid user experience that encourages powerful new ways of "thinking to code" and enable programmers **to see and understand their program executions**. Unfortunately current solutions for live exploration of scientific computing such as **Jupyter notebooks**, only provide a limited form of remote REPLs, with no support for a full development experience (browsers, debuggers) or domain-specific needs.

Domain-Specific Languages (DSL) offer through appropriate notations and abstractions, expressive power focused on, and usually restricted to a particular problem domain. DSL for scientific modelling & simulation allow experts to express their models in various ways: equations (ODEs, PDEs, ...), discrete events, processes, agents, etc ... A domain-specific language instead of describing the domain of computation itself (expressions, statements, variables, methods, functions, classes, ...) as do «General Purpose Languages» (like C/C++, Java, Python etc.), re-uses the formalisms, vocabulary and notions of the domain it describes.

APPROACH

Our goal is to be able to combine live programming and domain-specific languages (DSL) in the context of modelling and simulation for Computational Science. **To achieve this goal we need to adapt and reuse existing solutions to both live-programming & domain-specific requirements. At its core our envisioned solution should allow for the manipulation of heterogeneous (CPU/GPU) task graphs at runtime for reaching the needed interactivity and performance, taking advantage of established compilation platforms such as LLVM and runtimes such as StarPU.** On top of this live HPC infrastructure, will build **Live Tools** that support the execution of a remote target used for live runtime editing, instrumentation and analysis. More specifically we would like to investigate the applicability of **omniscient time-travelling debugging** that allows users to go both back and forth in computational time, to understand the consequences of code/data modifications and to give suitable feedback to the experts. To fulfil the requirement of liveliness in the context of computational science, we also want to investigate **surrogate models** and **anytime algorithms**. We have already made some early experiments in the context of live epidemiological simulation where an ODE solver is dynamically selected depending of properties of the model. We want to be able to upscale these experiments with the suitable HPC infrastructure.

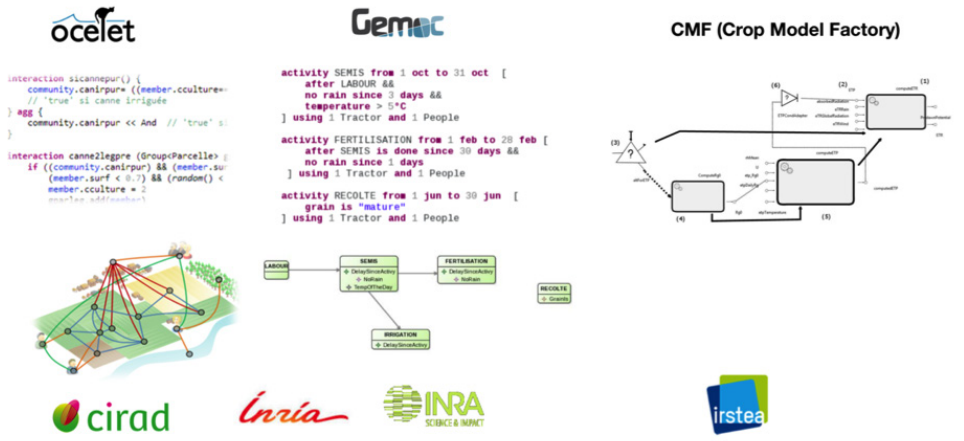

Fig. 2: DSL for Computational Science

CASES STUDIES

Provided by UMMISCO: Mathematical modelling of infectious diseases often uses simulation models to explore transmission mechanisms and to plan potential management strategies to control the epidemics. UMMISCO developed a DSL called **KENDRICK** that allows epidemiologists who are not CS experts to build/analyze their models very easily through composable and reusable epidemiological concerns (control strategies, strains, propagation mechanisms, spatial aspects). In the context of HPC for modelling and simulation, we want to be able to dynamically compose these epidemiological concerns in order to see the consequences of adaptation strategies in disease propagations. We already have some early experiment with a live epidemiological editor that allows us to see this composition, but without an HPC solution such as the one described in this proposal we are not able to scale our framework to bigger and more realistic models.

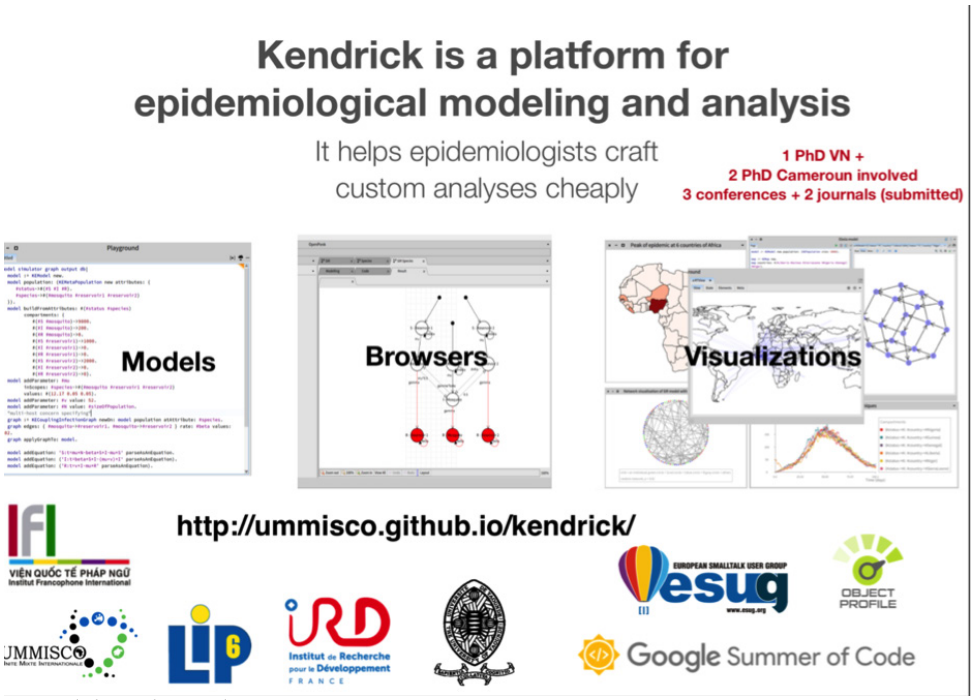

Fig. 3: Kendrick: example case studies

Provided by G-EAU: Well known modeling and simulation platforms are available for hydrological processes. However, before they can produce relevant prospective scenarios, several improvements are required, in particular accounting for feedback loops between social and physical dynamics. Distributed hydrological modeling enables the simulation of hydrological dynamics considering spatial heterogeneity of the catchments and consequently gives the opportunity to imagine local feedback loops related to human activities. We want to use an existing distributed hydrological model developed at IRSTEA for the Rhone River and integrate it with models of agricultural water uses. Agronomists and hydrologists of Irstea work together to define water users' behaviours and variables of interest and to relate them to state variables and outputs of the hydrological model in order to simulate two subcatchments of the Rhone River in an integrative model.