



www.legato-project.eu

Integrating OmpSs@FPGA within Eclipse

Presentation for EclipseCon 2019

Ruben Cano-Díaz and Xavier Martorell

Barcelona Supercomputing Center



The LEGaTO project has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No 780681

15/07/2019

OmpSs@FPGA

- Porting algorithms to IP cores in FPGA

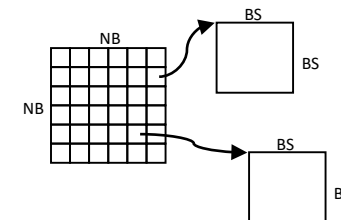
```
#define BS 128

void matrix_multiply(float a[BS][BS], float b[BS][BS],float c[BS][BS])
{
// matrix multiplication of two A, B matrices, to accumulate the result on C
  for (int ia = 0; ia < BS; ++ia)
    for (int ib = 0; ib < BS; ++ib) {
      float sum = 0;
      for (int ic = 0; ic < BS; ++ic)
        sum += a[ia][ic] * b[ic][ib];
      c[ia][ib] += sum;
    }
}
```

Kernel

```
...
for (i=0; i<NB; i++)
  for (j=0; j<NB; j++)
    for (k=0; k<NB; k++)
      matrix_multiply(A[i][k], B[k][j], C[i][j]);
...
```

Main program



OmpSs@FPGA

- Challenge
 - Multiple types of FPGA exist
- Solution
 - AutoVivado

```
#define BS 128

void matrix_multiply(float a[BS][BS], float b[BS][BS],float c[BS][BS])
{
#pragma HLS inline
    int const FACTOR = BS/2;
#pragma HLS array_partition variable=a block factor=FACTOR dim=2
#pragma HLS array_partition variable=b block factor=FACTOR dim=1
    // matrix multiplication of a A*B matrix
    for (int ia = 0; ia < BS; ++ia)
        for (int ib = 0; ib < BS; ++ib) {
#pragma HLS PIPELINE II=1
            float sum = 0;
            for (int id = 0; id < BS; ++id)
                sum += a[ia][id] * b[id][ib];
            c[ia][ib] += sum;
        }
}
```

Zynq-7000 Family
2x Cortex-A9 cores + FPGA
(32-bit platforms)



SECO AXIOM Board
Zynq U+ XCZU9EG-ES2



Trenz Electronics Zynq U+
TE0808 XCZU9EG-ES1

4x Cortex-A53 cores +
FPGA (64-bit platforms)



Alpha-Data
Alveo

Xilinx ZCU102

Outline

- Introduction to OmpSs@FPGA
- OmpSs integration within Eclipse / Eclipse Che
 - Autocompletion of directives
 - OmpSs toolchain on Docker Containers
- OmpSs@FPGA toolchain
- Evaluation & tracing
- Conclusions & future work

Autocompletion of directives

OmpSs/ OpenMP support in Eclipse

- Support for OmpSs and OpenMP development in eclipse
 - Plugins developed
 - Support for most of the programming models directives and clauses
 - Including small help descriptions
 - Based on context, with autocompletion
- Integration of the compilation environment
 - Eclipse Che

Autocompletion of directives

```
*matmul.c
118 unsigned int const b2size = BSIZE*BSIZE;
119 unsigned int const msize = atoi(argv[1]);
120 unsigned int const m2size = msize*msize;
121 unsigned char const check = argc > 2 ? atoi(argv[2]) : 1;
122 if (msize%BSIZE != 0) {
123     fprintf(stderr, "ERROR:\t<matrix size> must be multiple of <block size>\n");
124     usage(argv[0]);
125     exit(1);
126 }
127
128 size_t s = m2size*sizeof(elem_t);
129 elem_t* a = (elem_t *) (malloc(s));
130 elem_t* b = (elem_t *) (malloc(s));
131 elem_t* c = (elem_t *) (malloc(s));
132 if (a == NULL || b == NULL || c == NULL) {
133     fprintf(stderr, "ERROR:\tCannot allocate memory for the matrices\n");
134     exit(1);
135 }
136
137 #if defined(TIMING_ALL)
138     double t_ini_start = wall_time();
139 #endif
140
141     for (unsigned int i = 0; i < m2size/b2size; i++) {
142         setBlock(&a[i*b2size], (elem_t)VAL_A + i);
143         setBlock(&b[i*b2size], (elem_t)VAL_B - i);
144         setBlock(&c[i*b2size], (elem_t)VAL_C);
145     }
146
147 #if defined(TIMING_ALL)
148     #pragma omp taskwait
149 #endif
150     double t_start = wall_time();
151
152     for (unsigned int i = 0; i < msize/BSIZE; i++) {
153 //NOTE: Assuming that the following task will be executed in a shared memory environment.
154 //    Otherwise, it must define the input and output data of child tasks.
155 #pragma omp t
156 {
157     for (un
158     unsig
159     for (
160     un
161     un
162     ma
163     }
164     }
165     #pragma
166 }
167 }
168
169 #pragma omp
170 double t_en
171
172 unsigned int check_ok = TRUE;
173 if (check) {
```

target
task
taskloop
taskwait
taskyield
teams
threadprivate

task

The programmer can specify a task using the **task** construct. This construct can appear inside any code block of the program, which will mark the following statement as a task.

- #pragma oss task [clauses]
- structured-block
- private(<list>)
- firstprivate(<list>)
- shared(<list>)
- depend(<type>: <memory-reference-list>)
- <depend-type>(<memory-reference-list>)
- priority(<expression>)
- cost(<expression>)
- if(<scalar-expression>)
- final(<scalar-expression>)
- label(<string>)

Press 'Ctrl+Space' to show Default Proposals

Outline

- stdlib.h
- unistd.h
- matmul.h
- matmul.f
- setBlock(
- checkBlo
- matmulB
- matmulB
- main(int,

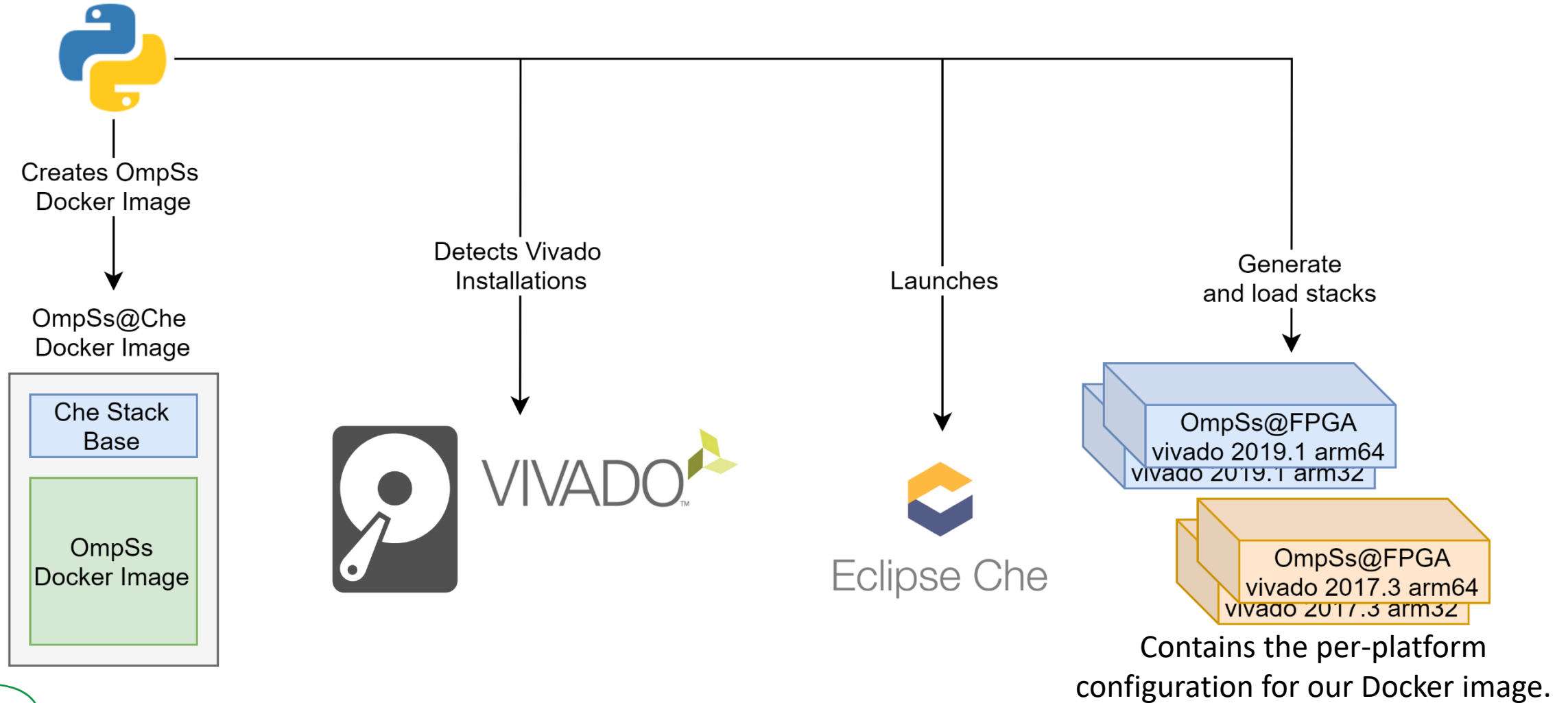
Compilation for the FPGA

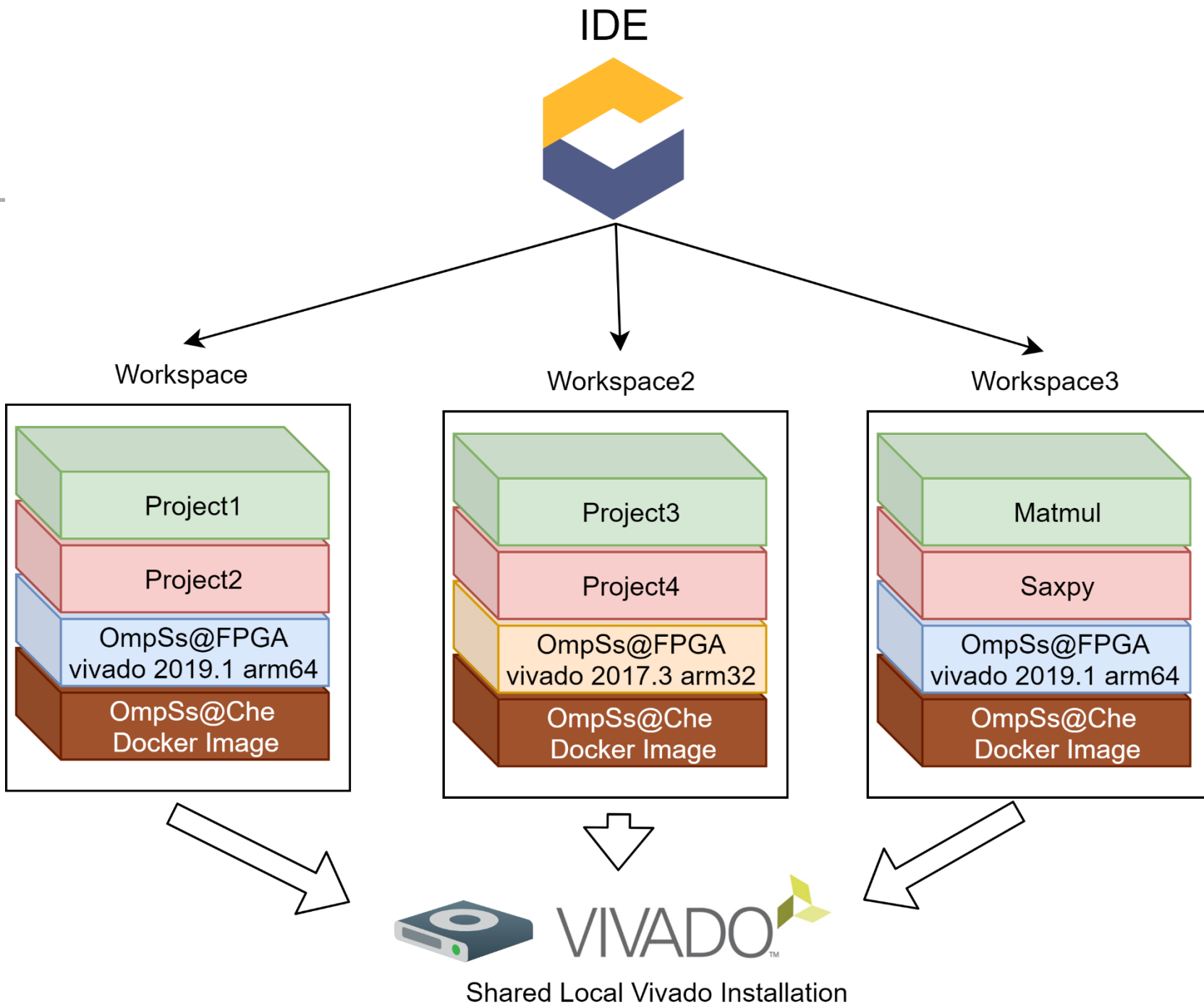
Incorporated to Eclipse Che

- Eclipse Che is the next version of the IDE environment
 - Easy administration on distributed environments
 - Suitable for local installation and server installation
 - Redesigned and per-stack customizable interface
 - **The compilation is done inside a Docker container**
- Our Containers
 - OmpSs compilation
 - With links to FPGA vendor tools
 - Xilinx Vivado
 - Automatic installation from git with a Python script

Compilation for the FPGA

Incorporated to Eclipse Che





All workspaces use our images and our stack configuration.

No configuration needed to compile for either platform thanks to our stack.

There can be any number of workspaces.

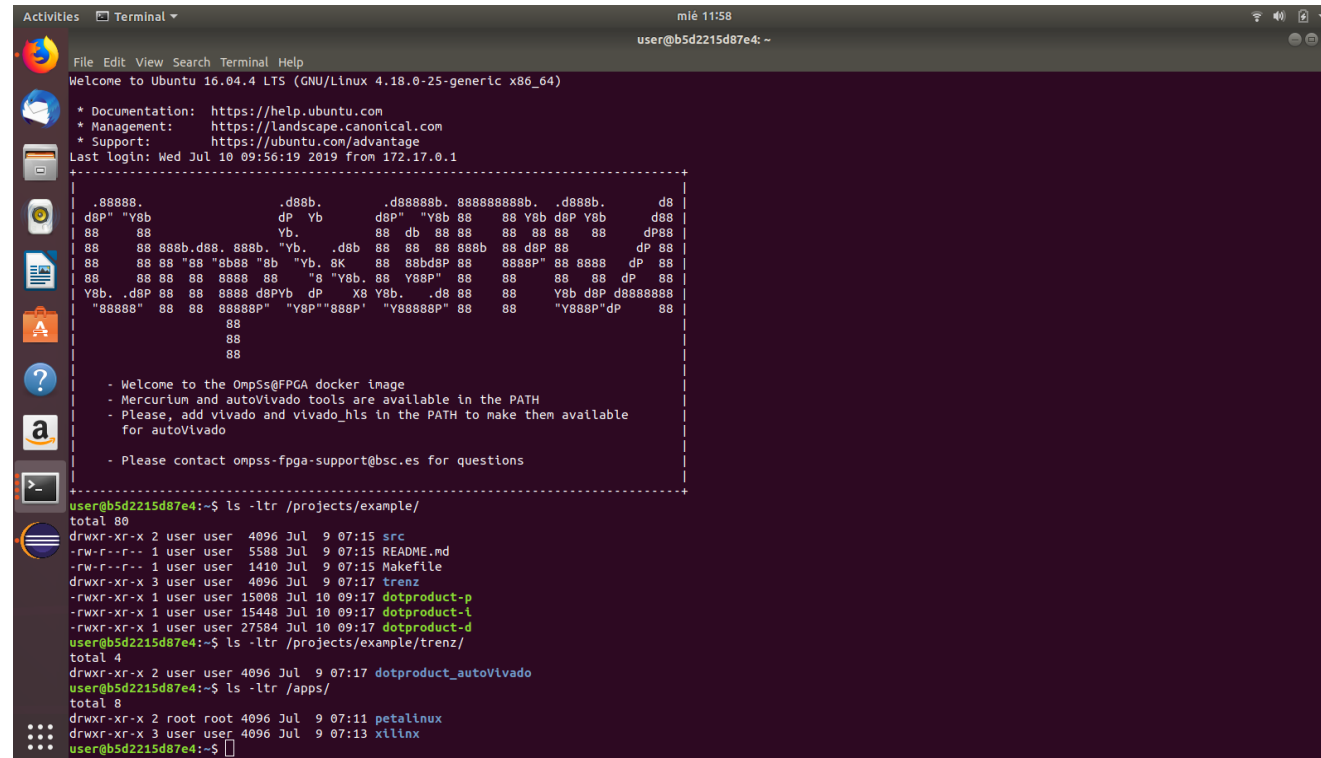
Each workspace contains its private projects.

The Vivado installation is shared between all workspaces.

Compilation for the FPGA

Incorporated to Eclipse Che

- Docker container
- Contains preconfigured OmpSs@fpga tools
 - Mercurium, Nanos, Xtasks, Xdma, Extrae, Papi, Cross Compiler
- Increases productivity and helps avoid configuration errors



```
mié 11:58
user@b5d2215d87e4: ~
File Edit View Search Terminal Help
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.18.0-25-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management:   https://landscape.canonical.com
* Support:      https://ubuntu.com/advantage
Last login: Wed Jul 10 09:56:19 2019 from 172.17.0.1

.,88888.,      .d88b.,      .d88888b., 888888888b., .d888b.,  d8
d8P" "Y8b      dP Yb  d8P" "Y8b 88 88 Y8b d8P Y8b  d88
88 88      Yb.      88 db 88 88 88 88 88 88  dP88
88 88 88b.d88. 888b. "yb. .d8b 88 88 88 888b 88 d8P 88  dP 88
88 88 88 "88 "8b88 "8b "yb. 8K 88 88bd8P 88 8888P" 88 8888  dP 88
88 88 88 88 8888 88 "8 "Y8b. 88 Y88P" 88 88 88 88 dP 88
Y8b. .d8P 88 88 8888 d8PYb dP X8 Y8b. .d8 88 88 Y8b d8P d8888888
"88888" 88 88 88888P" "Y8P""888P" "Y88888P" 88 88 "Y888P"dP 88
88
88
88

- Welcome to the OmpSs@FPGA docker image
- Mercurium and autoVivado tools are available in the PATH
- Please, add vivado and vivado_hls in the PATH to make them available
  for autoVivado
- Please contact ompss-fpga-support@bsc.es for questions

user@b5d2215d87e4:~$ ls -ltr /projects/example/
total 80
drwxr-xr-x 2 user user 4096 Jul  9 07:15 src
-rw-r--r-- 1 user user 5588 Jul  9 07:15 README.md
-rw-r--r-- 1 user user 1410 Jul  9 07:15 Makefile
drwxr-xr-x 3 user user 4096 Jul  9 07:17 trenz
-rwxr-xr-x 1 user user 15008 Jul 10 09:17 dotproduct-p
-rwxr-xr-x 1 user user 15448 Jul 10 09:17 dotproduct-l
-rwxr-xr-x 1 user user 27584 Jul 10 09:17 dotproduct-d
user@b5d2215d87e4:~$ ls -ltr /projects/example/trenz/
total 4
drwxr-xr-x 2 user user 4096 Jul  9 07:17 dotproduct_autoVivado
user@b5d2215d87e4:~$ ls -ltr /apps/
total 8
drwxr-xr-x 2 root root 4096 Jul  9 07:11 petalinux
drwxr-xr-x 3 user user 4096 Jul  9 07:13 xilinx
user@b5d2215d87e4:~$
```

Compilation for the FPGA

Workspaces

- Creation of a workspace
 - Custom stack selection

Eclipse Che

Dashboard
Workspaces
Stacks
Factories
Administration

New Workspace CREATE & OPEN

NAME ? Fpga-OmpSs-Vivado-2017.3 ✓

SELECT STACK ? All Quick Start Single Machine Multi Machine Filters + Add Stack 20 ×

NAME	RAM
OmpSs 32b 2017.3 Custom Stack to compile for arm32 with vivado 2017.3 fpga OmpSs Mercurium Vivado 2017.3	2 GB
OmpSs 64b 2017.3 Custom Stack to compile for arm64 with vivado 2017.3 fpga OmpSs Mercurium Vivado 2017.3	2 GB

RAM ? dev-machine cherfpga - 2 GB +

PROJECTS + Add or Import Project

CREATE & OPEN

Compilation for the FPGA

Workspaces

- Using the workspace
 - Usage is the same as most IDEs
 - Install new software on the workspace using linux commands on the terminal
 - Can debug and run remotely using serial port or ssh or any existing linux technology.

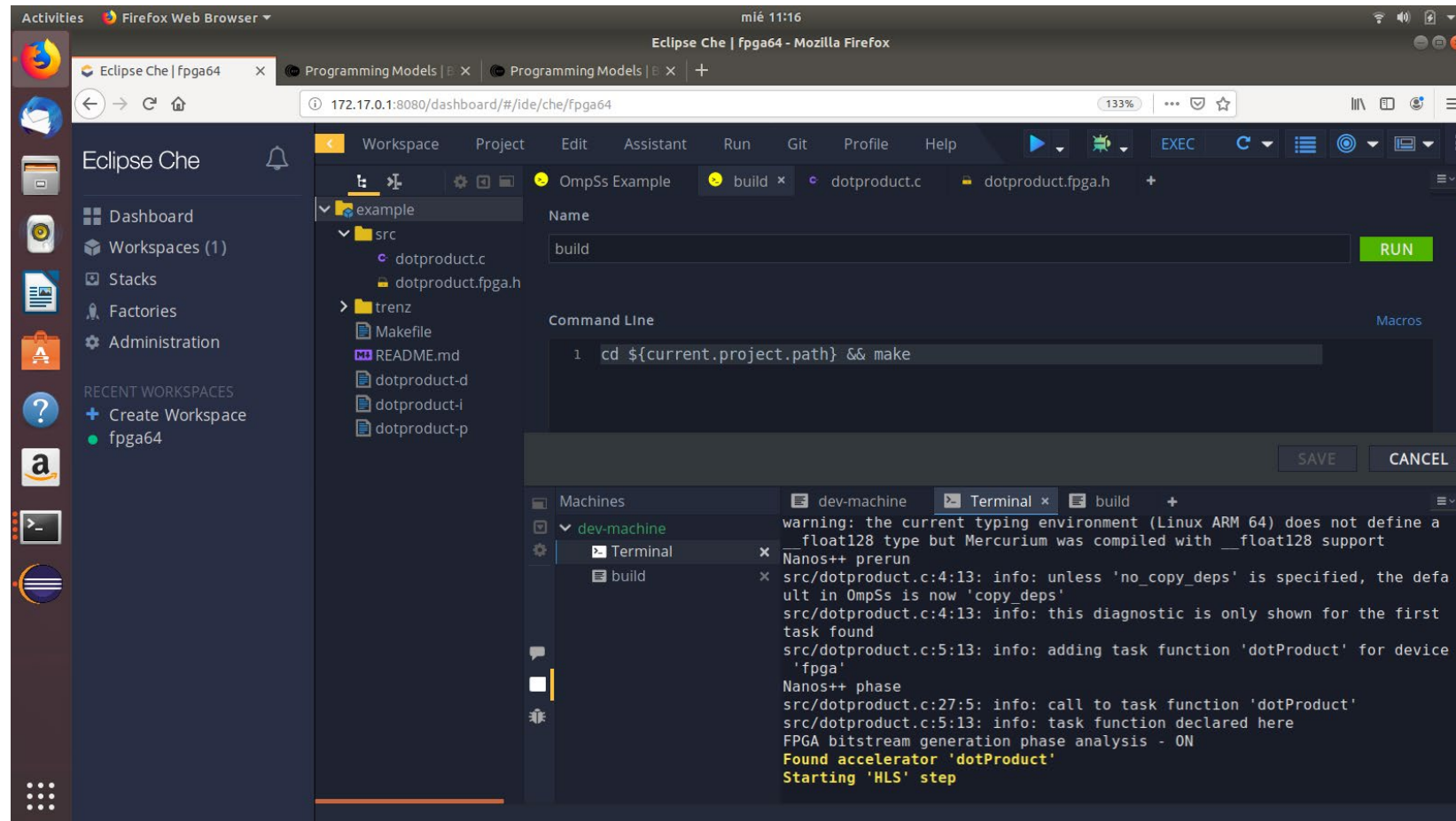
```
105     }
106     }
107 }
108 #endif
109 }
110 #endif
111
112 int main(int argc, char** argv) {
113     if (argc < 2) {
114         usage(argv[0]);
115         exit(1);
116     }
117
118     unsigned int const bsize = BSIZE*BSIZE;
119     unsigned int const msize = atoi(argv[1]);
120     unsigned int const m2size = msize*msize;
121     unsigned char const check = argc > 2 ? atoi(argv[2]) : 1;
122     if (msize*BSIZE != 0) {
123         fprintf(stderr, "ERROR:\t(matrix size) must be multiple of (block size)\n");
124         usage(argv[0]);
125         exit(1);
126     }
127
128     size_t s = m2size*sizeof(elem_t);
129     elem_t* a = (elem_t *) (malloc(s));
130     elem_t* b = (elem_t *) (malloc(s));
131     elem_t* c = (elem_t *) (malloc(s));
132     if (a == NULL || b == NULL || c == NULL) {
133         fprintf(stderr, "ERROR:\tcannot allocate memory for the matrices\n");
134         exit(1);
135     }
136
137     #if defined(TIMING_ALL)
138     double t_ini_start = wall_time();
139     #endif
140
141     for (unsigned int i = 0; i < m2size/bsize; i++) {
142         setBlock(&a[i*bsize], (elem_t)VAL_A + i);
143         setBlock(&b[i*bsize], (elem_t)VAL_B - i);
144         setBlock(&c[i*bsize], (elem_t)VAL_C);
145     }
146
147     #if defined(TIMING_ALL)
148     #pragma omp taskwait
149     #endif
150     double t_start = wall_time();
151
152     for (unsigned int i = 0; i < msize/BSIZE; i++) {
153         //NOTE: Assuming that the following task will be executed in a shared memory environment.
154         //      Otherwise, it must define the input and output data of child tasks.
```

Compilation for the FPGA

Compilation environment

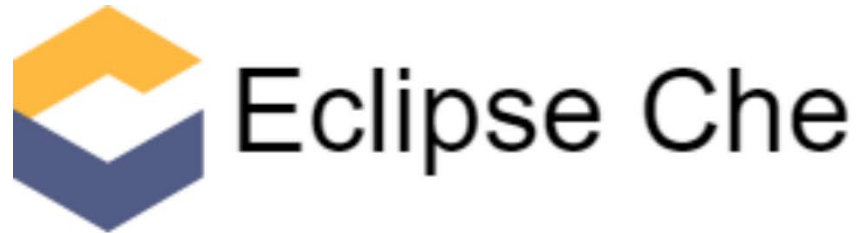
- Using the workspace

- Usage is the same as most IDEs
- Install new software on the workspace using linux commands on the terminal
- Can debug and run remotely using serial port or ssh or any existing linux technology.



Current developments

- Integration with Theia



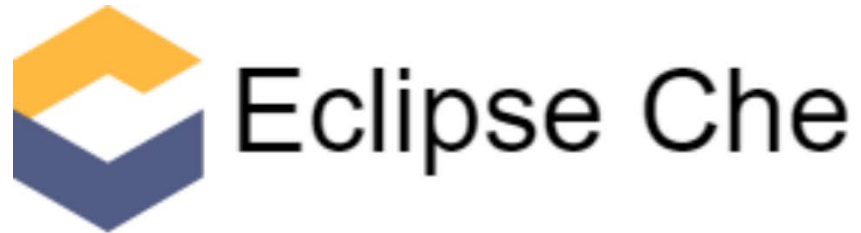
Cloud & Desktop IDE Platform



- Theia is an IDE that runs inside an Eclipse Che workspace
- When version 7 of eclipse Che releases, will be che's default IDE
- Is based on the opensource Monaco Editor that Powers Visual Studio Code
- It has compatibility with Visual Studio Code Plugins and Extensions

Current developments

- Integration with Theia



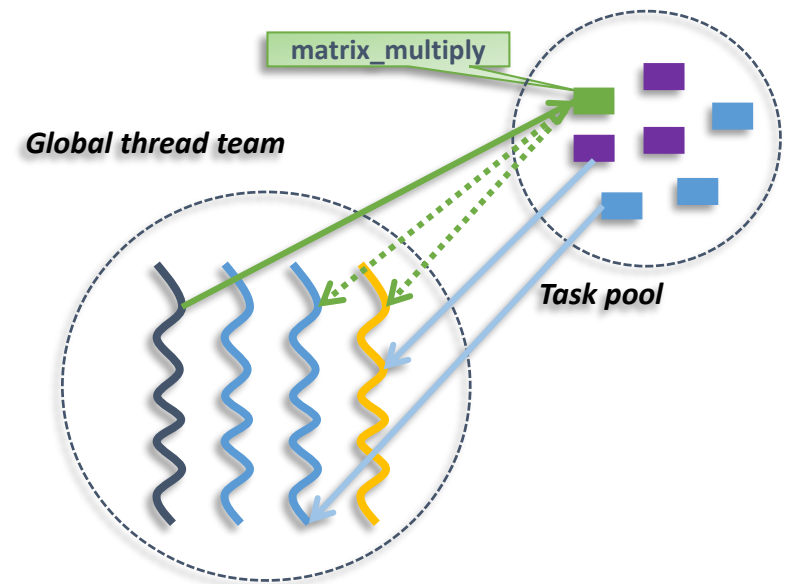
Cloud & Desktop IDE Platform



- Developed plugins Will be compatible with:
 - Eclipse Che + Theia
 - Visual Studio Code
- Can exploit all the features of the plugin platform
- The programmer can install optional or preferred plugins

Execution environment

- Nanos runtime system
- Resources available
 - OMP_NUM_THREADS / NX_GPUS / NX_NUM_FPGAS
 - Master and workers execute SMP tasks
 - One representative thread per accelerator
 - CUDA/OpenCL/FPGA device
- Getting work from the task pool
 - Different scheduling policies
 - “Implements” policy allows to exploit parallelism on “all resources”

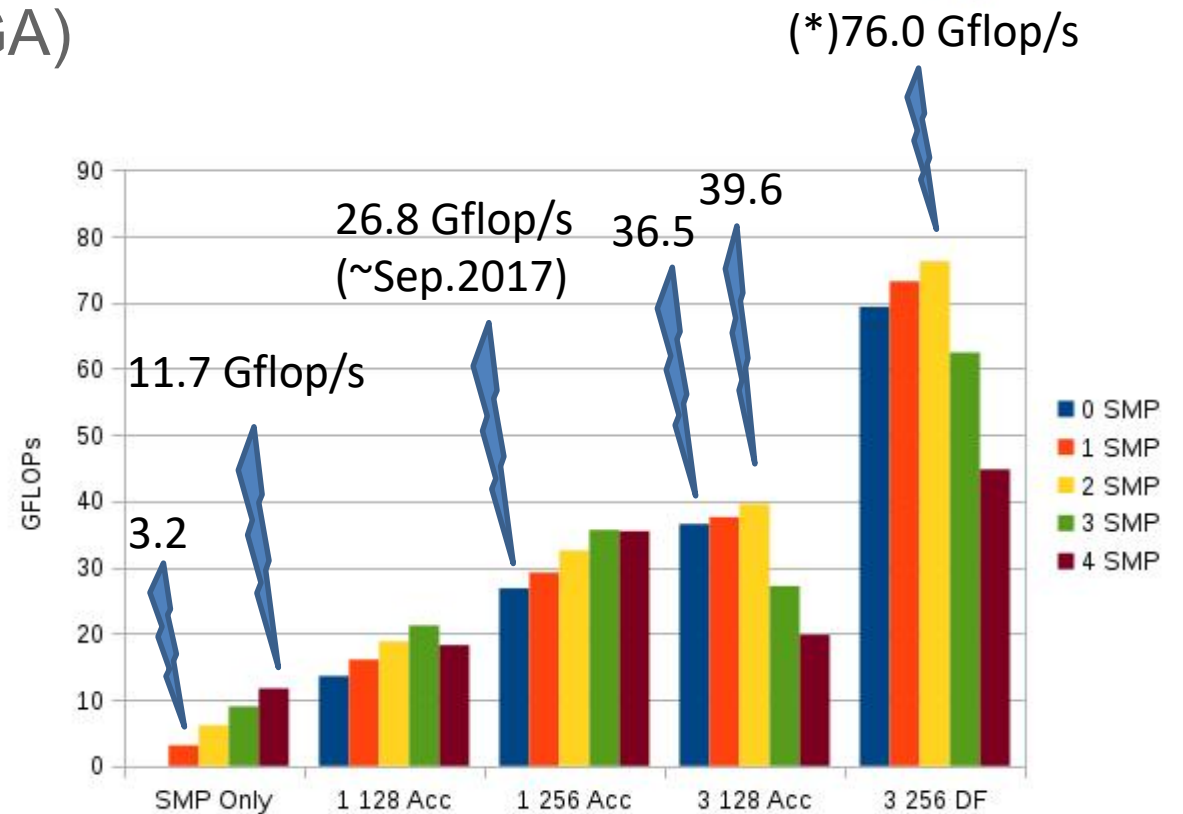


Execution environment

- Nanos runtime system

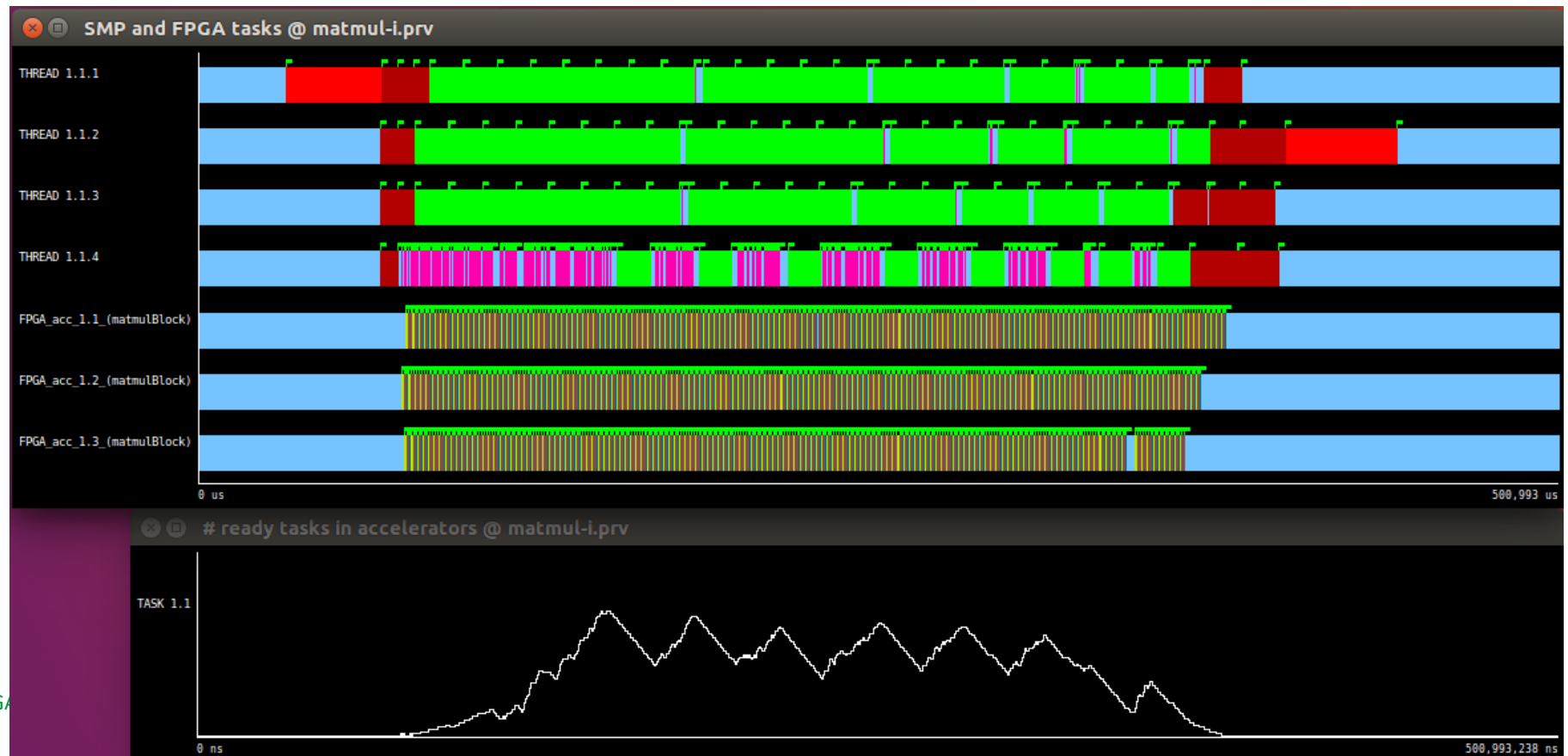


- Matrix multiplication 2048x2048 single precision
 - On Xilinx Zynq U+ (4x A53 + FPGA)
- Implements
 - SMP OpenBLAS
 - autoVivado on FPGA
 - Great success
- 300 MHz on FPGA
 - Outperforms cores
- Parallel task creation
- New dataflow IP core



Execution environment

- Tracing facilities
- 3x256 dataflow IP cores with “implements” and 4 ARM workers
 - ARM cores do a lot of work (on their possibilities)



Summary

Conclusions

- Developed IDE plugin for Eclipse for directives support
- Developed docker for automatic compilation on Eclipse Che
- Easy distribution of the OmpSs toolchain
- Use in Cloud environments

Summary

Future plans

- Keep improving the Eclipse/Eclipse Che support
- Keep improving the code generation with OmpSs toolchain
- Evaluating the possibility to use LLVM