

Grant Agreement:	604102	Project Title:	Human Brain Project
------------------	--------	----------------	---------------------

Document Title:	Neuromorphic Computing Platform v1 (public platform release)
Document Filename:	SP9_D9.7.4_PlatformRelease_V0.17.docx
Deliverable Number:	D9.7.4
Deliverable Type:	Prototype
Work Package(s):	WP 9.1, 9.2, 9.3, 9.5
Dissemination Level:	PU
Planned Delivery Date:	M30 / 31 Mar 2016 (public Platform release at the end of the Ramp-Up Phase)
Actual Delivery Date:	M30 / 30 Mar 2016 (platform release event. 1 st Neuromorphic Computing Application workshop on 22 March 2016)

Authors:	Karlheinz MEIER, Steve FURBER, Andrew DAVISON, David LESTER, Eric MÜLLER, SP9 contributors
Compiling Editors:	
Contributors:	
Coordinator Review:	UHEI (P45): Martina SCHMALHOLZ (STC)
Editorial Review:	EPFL (P1): Lauren ORWIN

Abstract:	<p>This Deliverable describes the v1 public release of the Neuromorphic Computing Platform as of the end of the Ramp-Up Phase.</p> <p>Both large scale systems, SpiNNaker (NM-MC1, in Manchester) and BrainScaleS (NM-PM1, in Heidelberg), are accessible remotely via the NMPI (integrated into the HBP Collaboratory) and reached their planned size of 500.000 cores for the SpiNNaker and 20 wafer scale systems for the BrainScaleS machine.</p> <p>A public live streamed release event (“1st Neuromorphic Computing Application Workshop”) with max. 100 concurrent viewers on YouTube took place on 22 March 2016.</p> <p>Access and info: http://neuromorphic.eu</p>
Keywords:	Neuromorphic Computing Platform
Available at:	www.humanbrainproject.eu/ec-deliverables

Table of Contents

1. The Aim of this Document	4
2. How to Access the Neuromorphic Computing Platform	4
3. Platform User Instructions.....	4
4. Platform Testing and Quality Strategy	4
4.1 Web-Services	5
4.2 BrainScaleS (NM-PM).....	5
4.2.1 Quality	6
4.3 SpiNNaker (NM-MC)	7
5. Platform User Adoption Strategy	8
6. Help and User Feedback	9
6.1 User Feedback Received Month 18 - Month 30 (examples).....	9
6.2 Usage	9
7. Annex A: Platform Architectural Diagram	11
7.1 Architecture overview: NM-PI part.....	11
7.2 Architecture overviews: SpiNNaker part (NM-MC)	12
7.3 Architecture overview: BrainScaleS (NM-PM).....	14
8. Annex B: Software and Services Included in this Platform Release	20
9. Annex C: Summary - Platform Use Case Status.....	23
9.1.1 SP9NMP-UC-001: A single run of a simple network model	23
9.1.2 SP9NMP-UC-002: A scripted run of a complex network model with input data and parameter files	24
9.1.3 SP9NMP-UC-003: Using the Neuromorphic Computing Platform through the Collaboratory and Brain Simulation Platform	25
9.1.4 SP9NMP-UC-004: Parameter sweeps	26
9.1.5 SP9NMP-UC-005: Closed-loop experiment involving a virtual environment.....	27
9.2 Missing features / under development features	27
10. Annex D: Summary - Service IT Resource Planning	28
11. Annex E: Summary - Service Technology Readiness Levels (TRLs) Metrics	29
12. Annex F: Backlog / Bug-tracking	30
13. Annex G: IPR Status, Ownership and Innovation Potential	32

List of Figures and Tables

Figure 1: Screenshot of the Jenkins overview page	6
Figure 2: Screenshot of the Gerrit code review system	7
Figure 3: Screenshots as of 20 March 2016 of the Neuromorphic Computing Platform Dashboard (https://www.hbpneuromorphic.eu/dashboard/): Number of completed jobs on the NM computing systems BrainScaleS (NM-PM1) and SpiNNaker (NM-MC1)...	10
Figure 4: Software architecture diagram: NMPI part (web interface from Collaboratory to the platform system batch queues)	11
Figure 5: SpiNNaker software architecture diagram: Python software stack	12
Figure 6: SpiNNaker software architecture diagram: C software stack	12
Figure 7: SpiNNaker software architecture diagram: Remote access	13
Figure 8: Five SpiNNaker system racks (500.000 cores), fully wired, in the machine room in Manchester, 25 March 2016	13
Figure 9: Architecture diagram: Job submission for BrainScaleS (NM-PM1)	14
Figure 10: Architecture diagram: Software stack for NM-PM1 BrainScaleS jobs	15
Figure 11: Status of the HBP NM-PM1 system BrainScaleS as of 21 March 2016.....	16
Figure 12: The components of one BrainScaleS (NM-PM1) wafer module	16
Figure 13: BrainScaleS (NM-PM1) module from the power-board side.....	17
Figure 14: NM-PM1 BrainScaleS module with three of the four boards with FPGAs added..	17
Figure 15: Fully assembled NM-PM1 BrainScaleS module, FPGA-side	18



Table 1: Component list for the 20 NM-PM1 BrainScaleS systems	18
---	----

1. The Aim of this Document

This document describes the status of the Neuromorphic Computing Platform NM-MC1 - SpiNNaker, NM-PM1 - BrainScaleS and the related access software at the end of the ramp-up phase for the 30 March 2016 HBP platform release. On 22 March 2016 both large scale machines have been presented publicly in the 1st Neuromorphic Computing Application Workshop - streamed live on YouTube¹.

This Deliverable is also the documentation for reaching Milestone MS187 “Platform ready for community release“.

2. How to Access the Neuromorphic Computing Platform

The Neuromorphic Platform is one of six ITC Platforms that comprise the HBP Scientific Research Infrastructure. All these Platforms can be accessed via the HBP Collaboratory web interface:

<https://collab.humanbrainproject.eu/#/collab/19/nav/403>

Direct link to the Neuromorphic Computing Platform on the Collaboratory:

<https://collab.humanbrainproject.eu/#/collab/51/nav/244>

3. Platform User Instructions

The living document HBP Neuromorphic Computing Platform guidebook is accessible:

- From within the collaboratory at <https://collab.humanbrainproject.eu/#/collab/51/nav/1069>
- Publicly at <http://electronicvisions.github.io/hbp-sp9-guidebook/>

The Platform Documentation links are also collected in the separate Deliverable D9.7.5.

Info: D9.7.5 was also scheduled to include a roadmap describing plans for future Platform development, but this topic is covered in this document - see Annex F: Backlog (remaining bugs and new features to be added).

4. Platform Testing and Quality Strategy

The platform quality strategy is to use:

- Code review
- Robust software design (e.g. strictly enforced modularity with clearly defined interfaces)
- Version control
- Both automated and manual testing
- Continuous integration
- Environment consistency and isolation

¹ The workshop had up to 100 concurrent viewers on YouTube during the live event (plus a few on AdobeConnect). The events recording remains accessible on YouTube at <https://www.youtube.com/watch?v=khRPnIDeklg> As of 28 March the page showed 590 views.

- An agile development methodology incorporating feedback from users and co-design projects with other HBP sub-projects.
- Application of coding and style standards for C, C++ and Python code.
- Comprehensive documentation.

All components of the platform have automated tests, both unit tests and integration tests, which are run automatically on code changes and/or on a scheduled basis using continuous integration tools (e.g. Jenkins or Travis CI). In contrast to conventional software development, the interactions between hardware and software also have to be tested. A large part of the test suite therefore implements tests involving the neuromorphic systems. All code is under version control, using Git. All code changes are reviewed by at least one person other than the author, using a structured workflow (Gerrit or Github pull requests). Documentation is rebuilt and published automatically when the source is updated.

Some details for the different platform parts, which are independently developed and maintained in Gif-sur-Yvette (Web-services, CNRS), Manchester (SpiNNaker, UMAN), Heidelberg (most parts of the BrainScaleS wafer module, UHEI) and Dresden (FPGA code used in the BrainScaleS system, TUD) can be found below.

4.1 Web-Services

The web services are implemented using the Django framework, while front-end applications are implemented using the AngularJS Javascript framework. Components are deployed in Docker containers, which enables differences between development, staging and production environments to be minimized. All components have both unit tests and integration tests. For Python/Django components we use the Python "unittest" module and the "nose" test runner. For AngularJS components we use the "Jasmine" test framework and the "Karma" test runner. Each release of the platform is also tested by hand before moving from staging to production. For each component there is an issue tracker. All code changes are reviewed by at least one person other than the author before merging to the stable branch. Continuous integration testing with Travis CI is used for the open-source components (e.g. PyNN). We do not yet use continuous integration for the non-open-source components; this is planned for the SGA1 phase of the project.

4.2 BrainScaleS (NM-PM)

The development of the NM-PM1 software stack is based on automated testing and code review. All software packages are automatically built by "Jenkins" on the local cluster, tests are executed and evaluated. Revisions that pass all tests are automatically installed as "nightlies" to a distributed file-system. Both, debug and release builds provide a fast test-debug cycle in case of emerging problems in the main development branch.

In contrast to conventional software development, the interactions between hardware and software have to be tested. This is why a large part of the test suite implements tests involving the neuromorphic system. Low-level regression tests run at least on a nightly basis and provide information of basic system behaviour. Higher-level tests, often written in the PyNN description language, provide a simple quality measure for "modelling performance".

A third application of the continuous integration framework are automated simulations of chip components. This use case is a mixture of the previous two cases. Depending on the detail level of the chip or circuit simulation, different aspects can be tested. Current work heads in the direction of automated evaluation of neuroscientific benchmarks on circuits that are still under development.

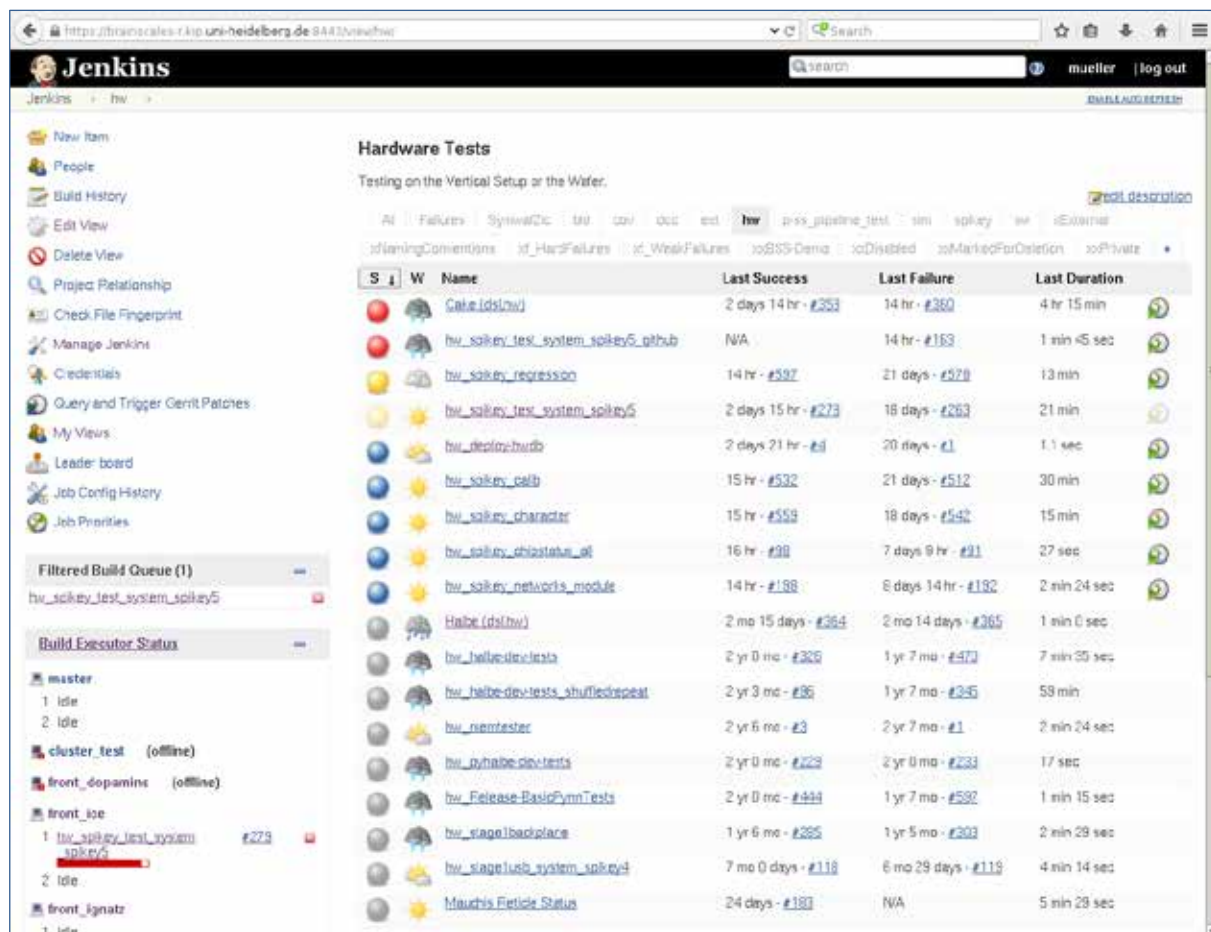
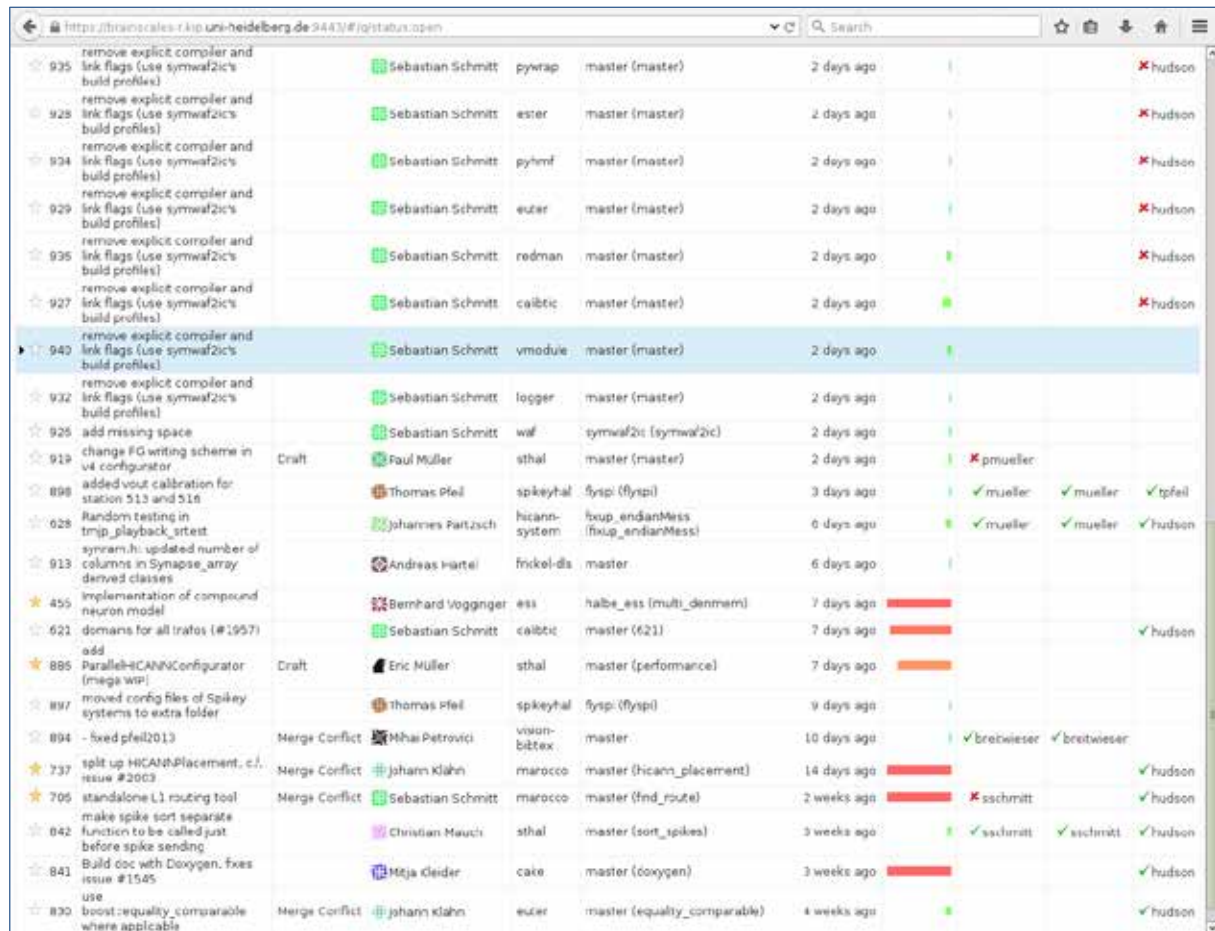


Figure 1: Screenshot of the Jenkins overview page

4.2.1 Quality

Another aspect of software development is code quality. The NM-PM1 software team migrated to the “gerrit” review software at the end of 2014. By now, all software core repositories and the major FPGA development repository have been moved into the code review system. As a general rule, code modifications may only be passed upstream if at least one other person tests and agrees with the proposed changes. Additionally, all proposed changes are test-built and tested by the previously mentioned “Jenkins” CI tool. Failing builds or tests yield a negative vote on the “changeset”.

However, robust software design is the main obligation when creating software for a new computing platform. As a consequence, the core software stack itself is written in a strongly-typed language, C++. Strictly type-safe APIs provide encapsulation of the different abstraction layers. The local code reviewers and our coding guidelines encourage modern C++ design. As an additional fast access to the software, many of our software layers provide an auto-generated Python-based interface which exposes the C++ APIs to the interpreted language. Thus, we allow non-expert programmers to access the hardware by tools that require very little expertise; in particular, many students use “Jupyter” for measuring and analysis scripts.



Issue	Summary	Author	Reviewer	Branch	Age	Status	Comments
935	remove explicit compiler and link flags (use symwaf2ic's build profiles)	Sebastian Schmitt	pywrap	master (master)	2 days ago	✓	hudson
929	remove explicit compiler and link flags (use symwaf2ic's build profiles)	Sebastian Schmitt	ester	master (master)	2 days ago	✓	hudson
934	remove explicit compiler and link flags (use symwaf2ic's build profiles)	Sebastian Schmitt	pyhmf	master (master)	2 days ago	✓	hudson
929	remove explicit compiler and link flags (use symwaf2ic's build profiles)	Sebastian Schmitt	euler	master (master)	2 days ago	✓	hudson
935	remove explicit compiler and link flags (use symwaf2ic's build profiles)	Sebastian Schmitt	redman	master (master)	2 days ago	✓	hudson
927	remove explicit compiler and link flags (use symwaf2ic's build profiles)	Sebastian Schmitt	colbitt	master (master)	2 days ago	✓	hudson
940	remove explicit compiler and link flags (use symwaf2ic's build profiles)	Sebastian Schmitt	vmodule	master (master)	2 days ago	✓	hudson
932	remove explicit compiler and link flags (use symwaf2ic's build profiles)	Sebastian Schmitt	logger	master (master)	2 days ago	✓	hudson
925	add missing space	Sebastian Schmitt	waf	symwaf2ic (symwaf2ic)	2 days ago	✓	hudson
919	change FG writing scheme in v4 configurator	Craft	Paul Müller	sthal	master (master)	2 days ago	✗ pmueller
898	added vout calibration for station 513 and 516	Thomas Pfeil	spikehal	flyspi (flyspi)	3 days ago	✓ mueller	✓ mueller
928	Random testing in tmp_playback_test	Johannes Partzsch	hccann-system	fixup_endianMess (fixup_endianMess)	6 days ago	✓ mueller	✓ mueller
913	synnam.h: updated number of columns in Synapse_array derived classes	Andreas Martel	frickel-dls	master	6 days ago	✓	hudson
455	Implementation of compound neuron model	Bernhard Vogginger	ess	halbe_ess (multi_denmem)	7 days ago	✗	hudson
621	domains for all Inafos (#1957)	Sebastian Schmitt	colbitt	master (621)	7 days ago	✓	hudson
885	ParallelHCANNConfigurator (mega WIP)	Craft	Eric Müller	sthal	master (performance)	7 days ago	✗
897	moved config files of Spike systems to extra folder	Thomas Pfeil	spikehal	flyspi (flyspi)	9 days ago	✓	hudson
894	- fixed Pfeil2013	Merge Conflict	Mihai Petrovici	vision-bibtex	master	10 days ago	✓ bretwieser
737	split up HCANNPlacement.c: issue #2003	Merge Conflict	Johann Klähn	marocco	master (hccann_placement)	14 days ago	✓ hudson
705	standalone L1 routing tool	Merge Conflict	Sebastian Schmitt	marocco	master (find_route)	2 weeks ago	✗ sschmitt
642	make spike sort separate function to be called just before spike sending	Christian Mauch	sthal	master (sort_spikes)	3 weeks ago	✓ sschmitt	✓ sschmitt
841	Build doc with Doxygen, fixes issue #1545	Mitja Kleider	cake	master (doxygen)	3 weeks ago	✓	hudson
830	boost: equality comparable where applicable	Merge Conflict	Johann Klähn	euler	master (equality_comparable)	4 weeks ago	✓ hudson

Figure 2: Screenshot of the gerrit code review system

4.3 SpiNNaker (NM-MC)

Software development for the SpiNNaker platform is all performed using git, with GitHub providing additional pull request and code review features. Changes to the software, including features and bug fixes, are developed within branches of the git repository. As the software interacts with the hardware, we constantly carry out testing during the development using the PyNN scripts found in the PyNNExamples git repository within the SpiNNakerManchester GitHub organization. Once a branch is believed to be working, a GitHub pull request is created. Someone in the software development team who did not make the changes will then look through the code, checking for PEP8 compliance in Python code, and against C coding standards for the SpiNNaker C code, as well as checking the changes themselves to be correctly implemented and that the flow of any new functionality makes sense. Before the pull request is merged into the master of the git repository, it is again tested against the PyNNExamples scripts to ensure that no changes have broken the functionality, as well as against a number of other integration tests that have been gathered from users, including the 10%-scaled version of the Potjans & Diesmann Microcortical Column model. We also have a growing number of unit tests for the code which are continuously tested using the Jenkins continuous integration suite. This ensures that the master branch of the repository contains only high quality and generally functional code. Finally, releases of the software are made once a number of new features have been added, or in time for training workshops. Before a release is generated, it is further tested using all gathered integration tests. The release is then tagged in the git repository and Python and built binaries are pushed into the PyPI repositories for distribution. This ensures that it is always possible to return to a previous version of the software should this be required by a particular script.

To ensure that the software is coherent, a fully enforced modular separation is maintained, with a stack hierarchy that ensures that modules only depend on those below them in the hierarchy. Interfaces between the layers make it easy to modify the internals of the modules with as little interference to the other modules as possible. Abstraction is used throughout the code to ensure that when a refactor of an interface is required, it is easy to find all the places that rely upon or implement the interface, so that modules don't end up broken due to the refactoring. We also make use of integrated development environments (Eclipse, PyDev and PyCharm) which aid in the development of the software through highlighting defects and style violations, and also making it easy to move and rename classes and interfaces when necessary.

5. Platform User Adoption Strategy

The Neuromorphic Computing Platform expects to attract users from three areas: Academia, industry and education. In academia, users come mostly from neuroscience and machine learning. Neuroscience users will exploit the simulation speed of the platform machines to bridge the many different time scales present in biological systems which is not possible with conventional high performance computers. The study of plasticity, learning and development is among the most interesting challenges of brain science as it is the aspect of self-organisation that makes biological brains different from traditional computers. The strong involvement of the Neuromorphic Computing Platform in the future co-design project CDP5 reflects this special strength of neuromorphic computing. In machine learning the neuromorphic platform offers the possibility to study the features and performance of novel computing architectures based on spiking neurons. Examples are spiking Boltzmann machines, stochastic computing or deep learning networks. The novel aspects in this research are in particular related to the implementation of local learning and the corresponding performance benefits in energy and time usage. Industrial users are expected to use neuromorphic computing for applications in processing of large, complex data sets, specifically detecting and predicting spatio-temporal patterns. The corresponding application areas are (among others) in telecommunication, finances, automotive, robotics and possibly entertainment. Industry users will also be able to derive special, low-cost, custom neuromorphic chips from the networks developed on the large machines. Finally, educational users will introduce the concepts of neuromorphic computing to the next generation of computer users ranging from high school to university students. The HBP Neuromorphic Computing Platform is already offering a substantial amount of teaching material and even small-scale hardware systems to address this very important user group.

The neuromorphic computing platform has already implemented and put into place strategies to attract new users from all areas described before into neuromorphic computing. Academic and educational users are introduced to the systems at various international schools and workshops, some of them are organized internally by the HBP education group. The first Neuromorphic Computing Application Workshops has been carried out on March, 22nd 2016 (as live streamed event on YouTube) and is planned to be repeated each year throughout the HBP project duration.

Additional information for the SpiNNaker (NM-MC1) system: the primary users are research teams (mainly academic, though some in national labs and industry) interested in modelling biological or artificial neural networks for scientific research, real-time robotics, or similar. Outreach is performed through conference presentations and demonstrations, workshops (e.g. Capo Caccia and Telluride) and open training events (in Manchester and at HBP events).

SpiNNaker is available via the HBP collaboratory (the 500,000-core system), through board loans (primarily 4-node 72-core boards) and sales (single or multiple 48-node 864-core boards). Currently we can make sales only to academic research labs, but we are renegotiating our supply-side contracts to extend this.

6. Help and User Feedback

To obtain help in using the platform, please start by checking the online user documentation at: <http://electronicvisions.github.io/hbp-sp9-guidebook/> (publicly accessible link). The guidebook is also available from within the Collaboratory at: <https://collab.humanbrainproject.eu/#/collab/51/nav/1069>

If you need personal assistance, want to provide feedback or contribute to the on-going development of the platform, please use the contacts described in the guidebook at: http://electronicvisions.github.io/hbp-sp9-guidebook/getting_help.html

6.1 User Feedback Received Month 18 - Month 30 (examples)

Early-access user feedback received (e.g. the SP11.3 partners used the system HBP internally):

- While the compatibility between the individual hardware PyNN interfaces saw constant improvement over the last half year, no platform implements the newest PyNN interface version (0.8) at this moment, requiring platform-specific workarounds (Spikey is at version 0.6 and NM-MC1/NM-PM1 are at 0.7). Once these incompatibilities are ironed out, the compute platform fulfills its task of providing user-friendly access to the neuromorphic hardware systems and allowing to run scripts on entirely different platforms at the flip of a switch.
- Suggestion for the job submission: allow direct upload of a project archive (instead of URL of a public accessible repository or direct cut&paste)
- Suggestion: improve feedback regarding the current job status (position in the queue, current status of the machines, streaming of the process output)
- Access to the NM-PM1/HICANN system generally works flawlessly. The system has -- in comparison to NM-MC1 -- small setup and extremely small execution times. However, the execution of larger networks does not provide meaningful results. This issue is being actively investigated at Heidelberg and will hopefully be resolved in the near future. (-> V4 Wafer)
- A limited number of neuron models have as yet been implemented on SpiNNaker (NM-MC-1). Preliminary work indicates that AdEx could be implemented in fixed point arithmetic, but Hodgkin-Huxley really needs floating point arithmetic. We intend to extend the number of models supported during SGA1.

6.2 Usage

A usage graph for the components of the Neuromorphic Computing Platform is available as a dashboard at: <https://www.hbpneuromorphic.eu/dashboard/>

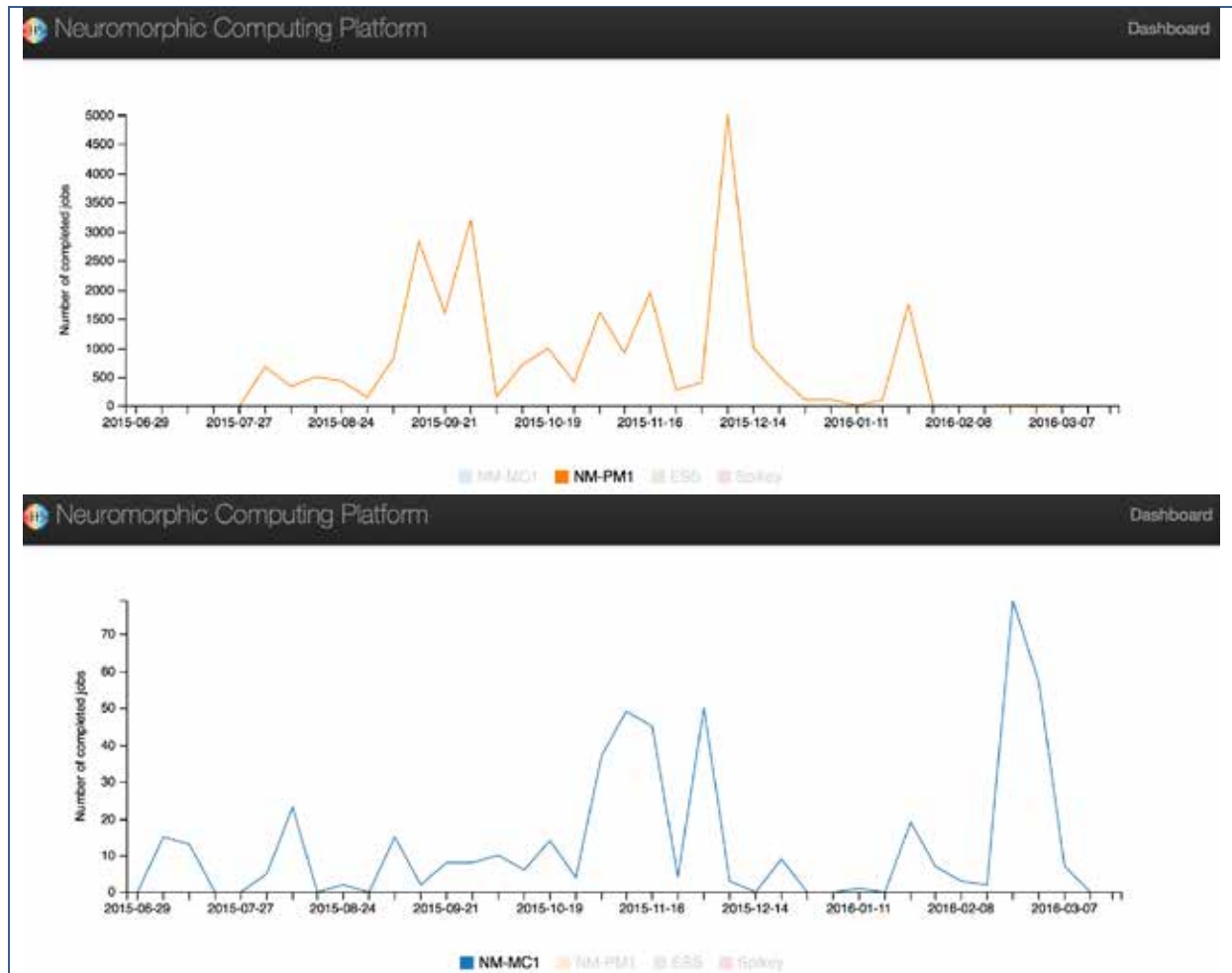


Figure 3: Screenshots as of 20 March 2016 of the Neuromorphic Computing Platform Dashboard (<https://www.hbpneuromorphic.eu/dashboard/>): Number of completed jobs on the NM computing systems BrainScaleS (NM-PM1) and SpiNNaker (NM-MC1)

7. Annex A: Platform Architectural Diagram

7.1 Architecture overview: NM-PI part

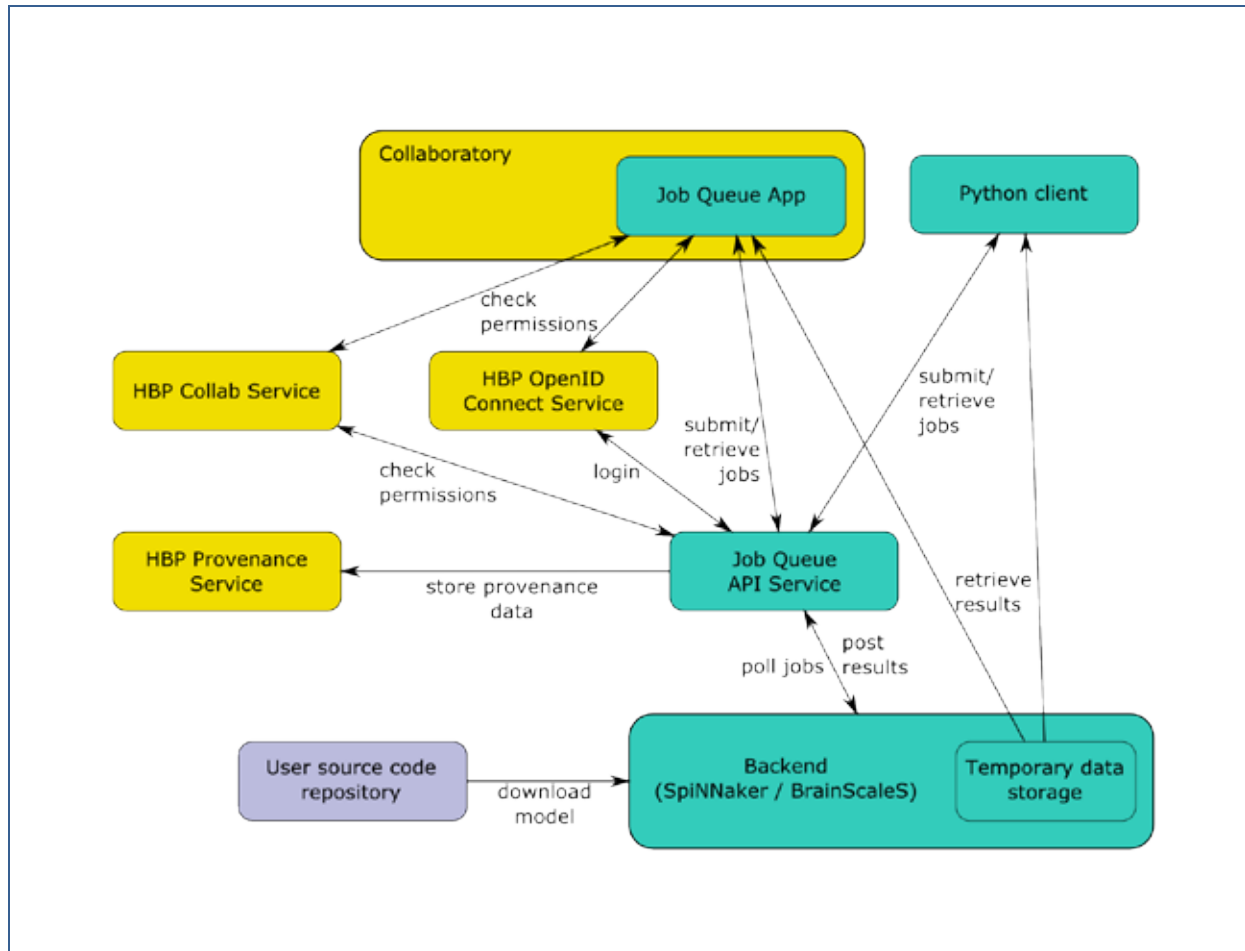


Figure 4: Software architecture diagram: NMPI part (web interface from Collaboratory to the platform system batch queues)

7.2 Architecture overviews: SpiNNaker part (NM-MC)

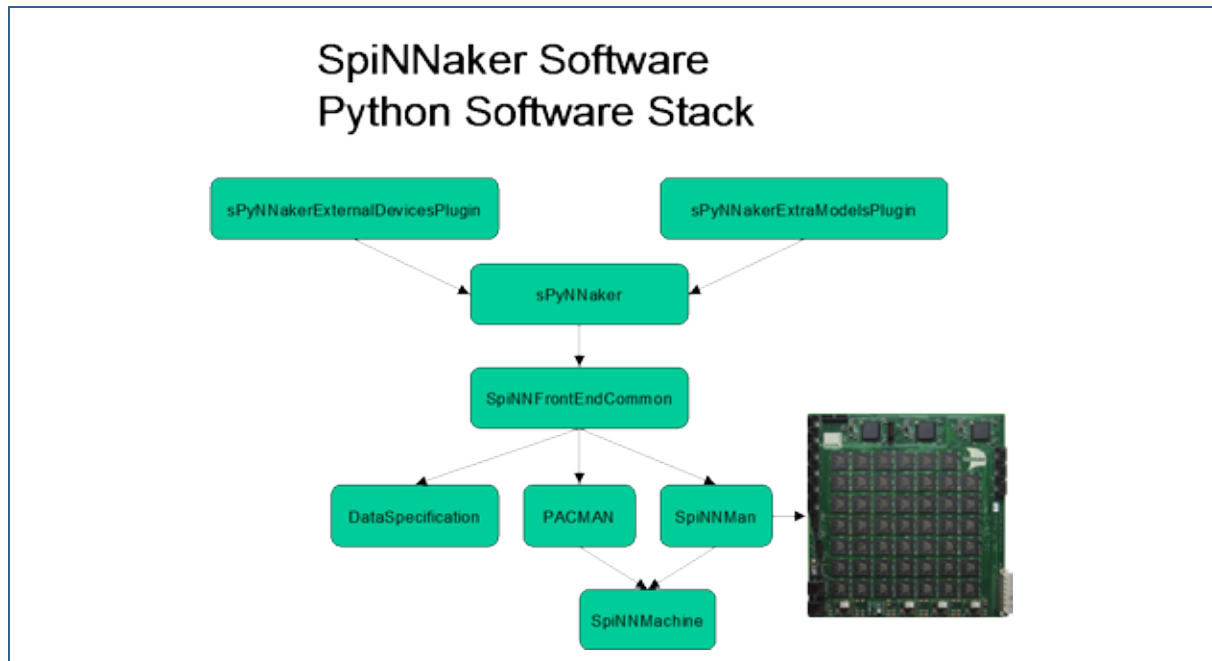


Figure 5: SpiNNaker software architecture diagram: Python software stack

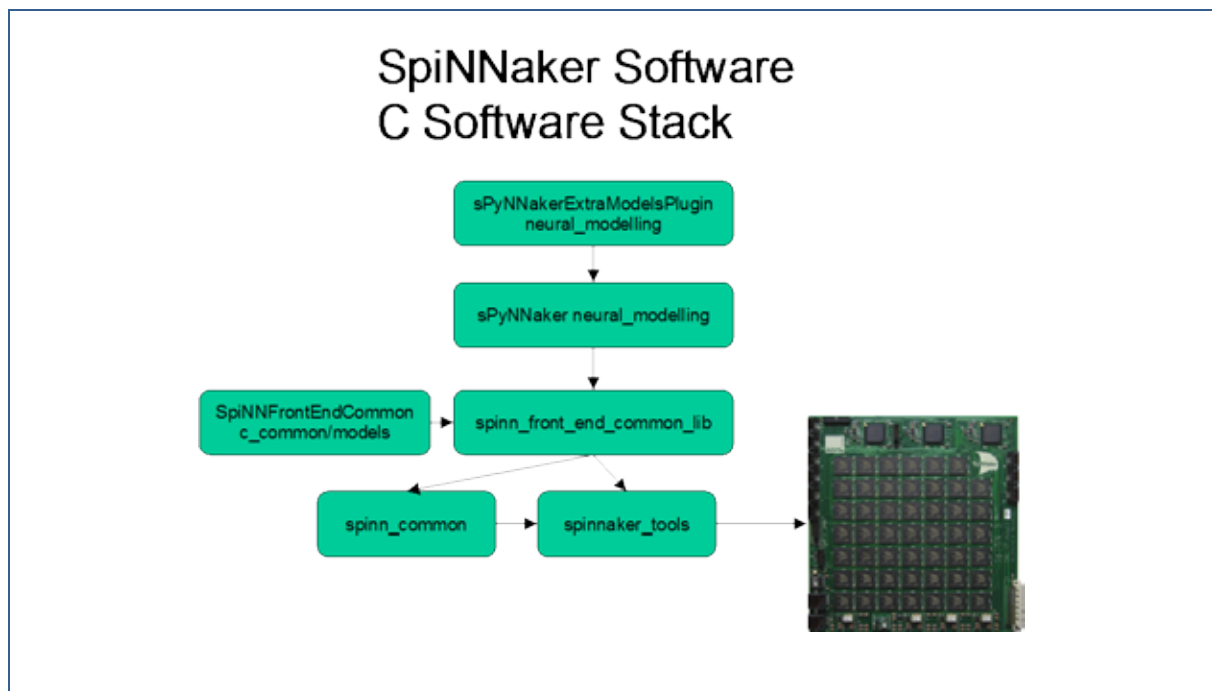


Figure 6: SpiNNaker software architecture diagram: C software stack

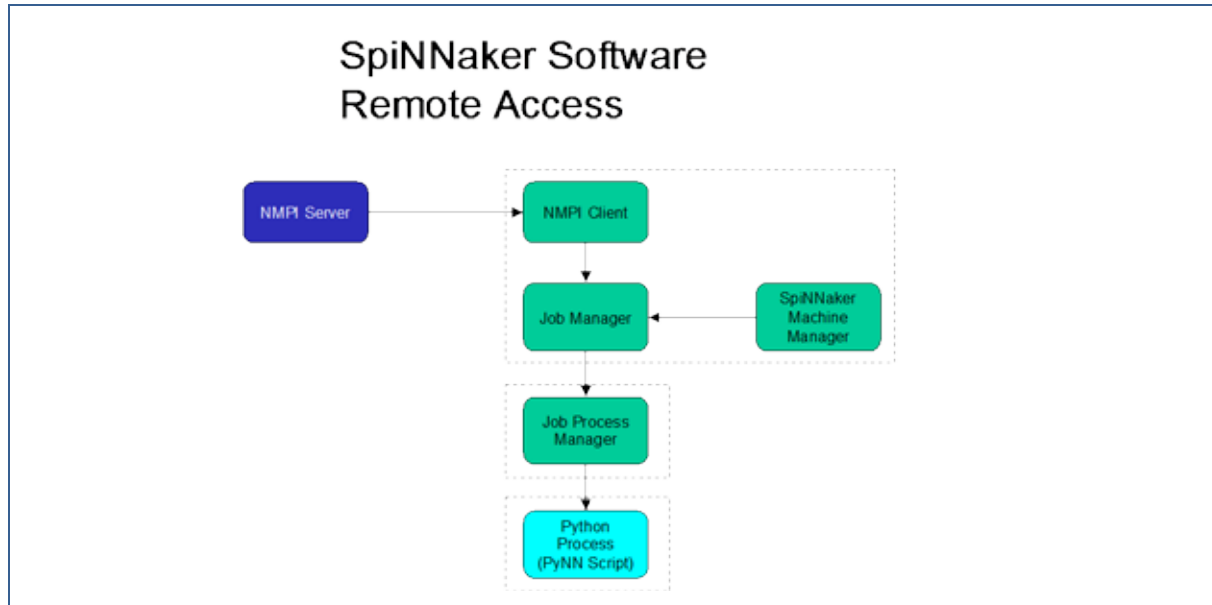


Figure 7: SpiNNaker software architecture diagram: Remote access

Finally, we have an image of the completed machine. This has five cabinets of five racks each, with each rack containing twenty-four 48 node boards. This machine thus comprises 28,800 SpiNNaker chips, or 518,400 individual ARM-968 cores. This machine constitutes Milestone MS178. Further extensions to this machine will be made in SGA-1.



Figure 8: Five SpiNNaker system racks (500.000 cores), fully wired, in the machine room in Manchester, 25 March 2016

7.3 Architecture overview: BrainScaleS (NM-PM)

Remote access to the NM-PM1 system is provided via NMPI. A client program accesses the NMPI Rest API, pulls new jobs and provides back updated job status information for previously pulled jobs. The NMPI client passes down jobs to the standard SLURM resource managing software framework. Neuromorphic hardware as well as cluster resources and other factors (e.g. fairness between users or jobs) are managed by the same software package. Hardware resources are allocated to individual jobs and, as soon as the requirements are satisfied, the job is executed on cluster nodes. A distributed cluster file-system is used to access job output data from an Apache-based https server, which provides user access to the result data.

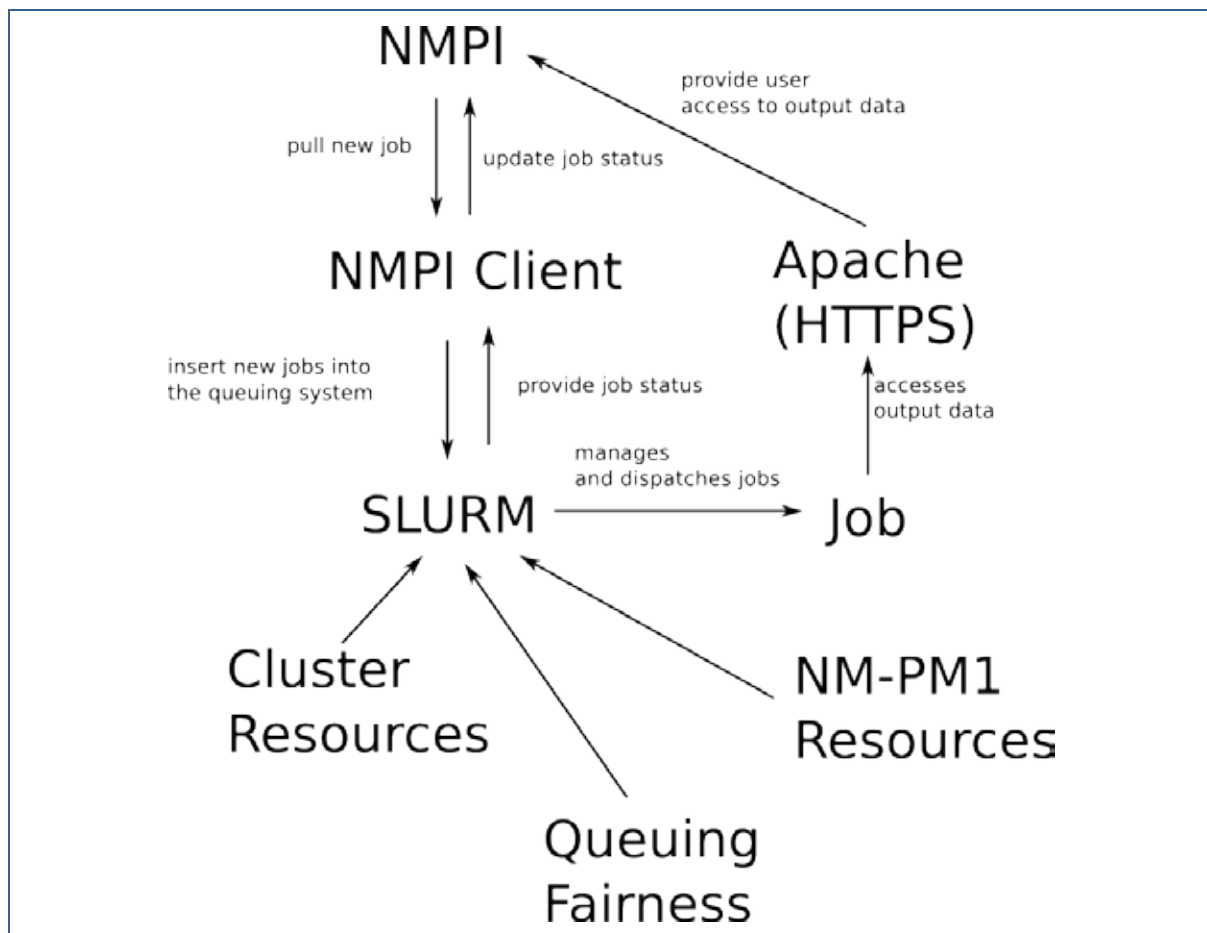


Figure 9: Architecture diagram: Job submission for BrainScaleS (NM-PM1)

The NM-PM1 software stack contains one main execution pipeline that starts from the PyNN-based network description (the PyNN implementation for the BrainScaleS system is called *PyHMF*). Additional software layers provide a more C++-friendly data representation of the mentioned neuronal network description. The translation from the PyNN biological description to a matching hardware configuration is performed by the map & route tool *Marocco*. Finally, a hardware access layer configures the system, controls input and records output data of experiments running on the neuromorphic system.

As all layers provide APIs, low-level tools can access individual layers. For instance, the calibration framework *cake* uses the low-level access layers to directly access single neuron circuits on the NM-PM1 system. For real-time/hybrid operation, a real-time-capable low-level API is provided by *VerCL*.

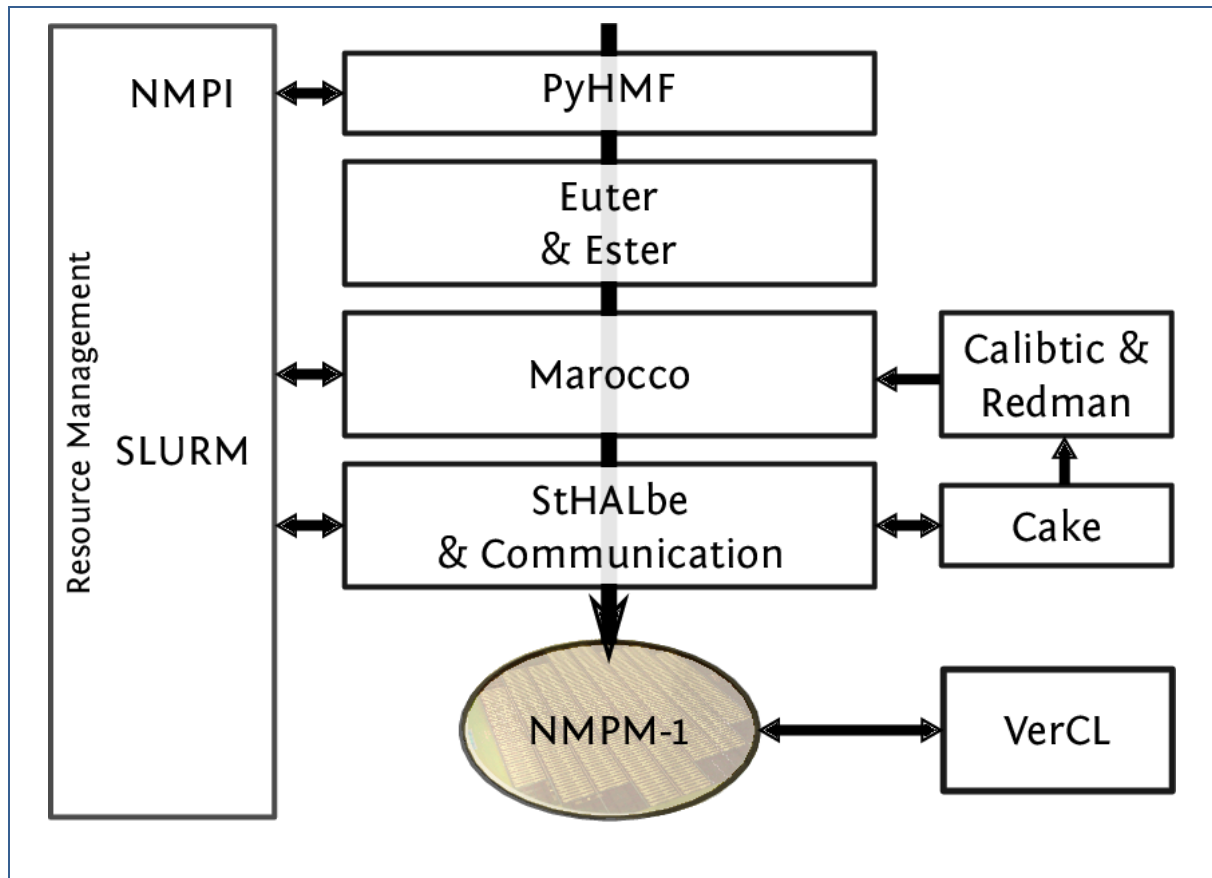


Figure 10: Architecture diagram: Software stack for NM-PM1 BrainScaleS jobs

Specification of the conventional compute cluster (4th rack from left, see Figure 11):

20 Compute Nodes: <ul style="list-style-type: none"> • Intel Core i7-4771 • 4 (+4HT) cores • 32GB • 2x10GbE uplink 	2 Frontend Servers: <ul style="list-style-type: none"> • Intel Xeon E5-2643v2 • 6 (+6HT) cores • 64GB • 2x40GbE uplink • 3TB SSDs 	1 (Slow) Storage Server <ul style="list-style-type: none"> • 64TB HDDs • 2x40GbE uplink
---	---	--

Summary:

- 92 real (+ 92 HT) cores
- 4.2TFlop/s (only the CPUs, GPUs not counted)
- 768GB RAM
- Wafer to node network supports 560 Gbit/s (this is still less than the maximal possible 20*48Gbit/s = 960Gbit/s from all wafers combined)
- Aggregated SSD streamed write supports ~5GB/s (i.e. the peak output of one single wafer module can be streamed to disk)
- 3TB (about 8 minutes at peak output of a single wafer) fast (SSD) and 50TB slow storage (RAID 6, 64 TB raw disk storage capacity)

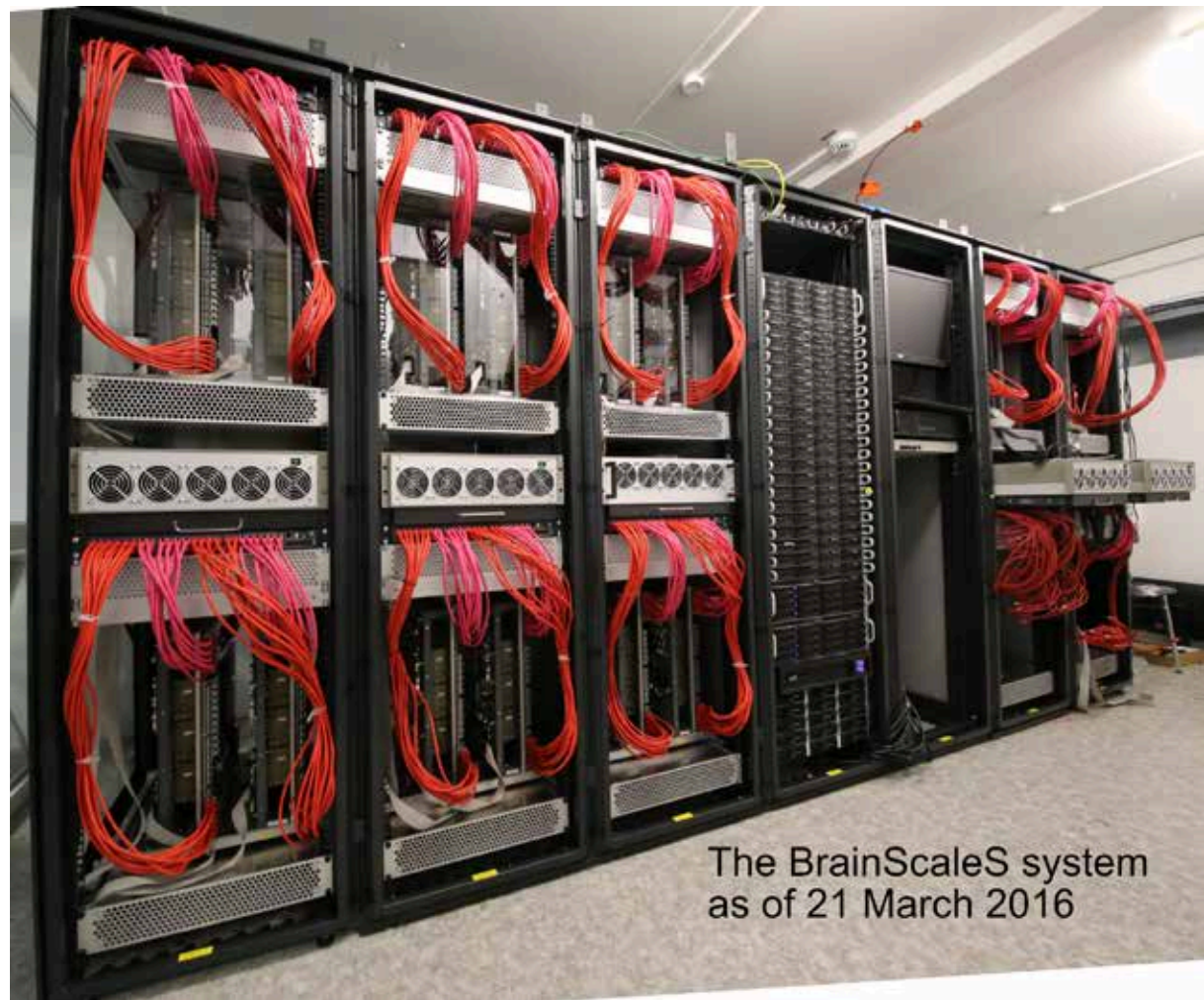


Figure 11: Status of the HBP NM-PM1 system BrainScaleS as of 21 March 2016
For the platform release day (30 March 2016) 20 systems are in place.

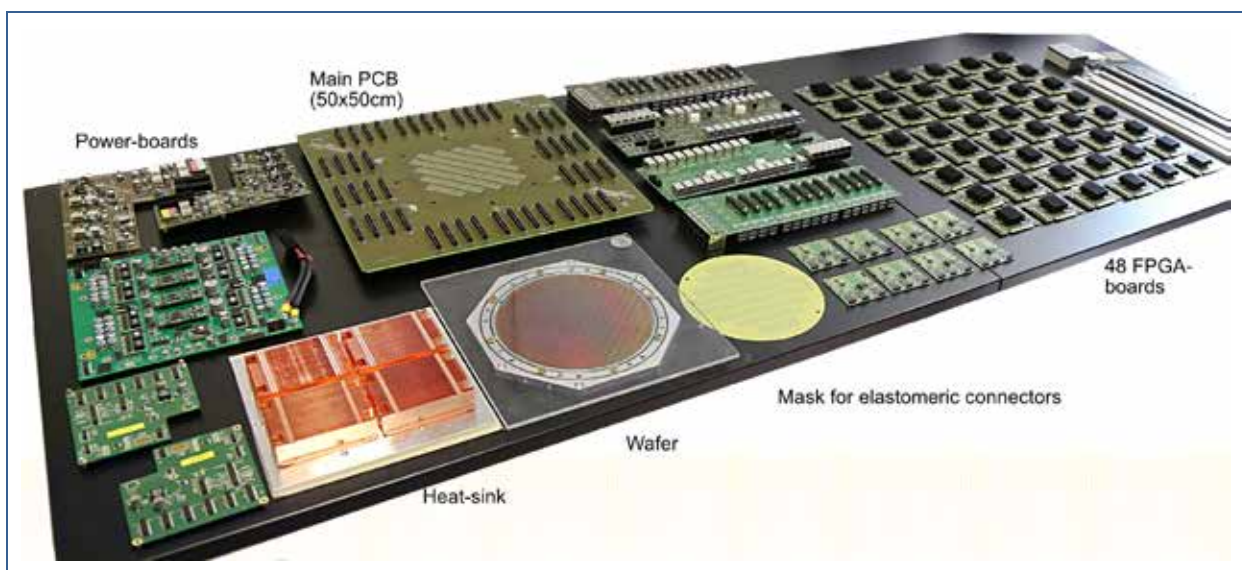


Figure 12: The components of one BrainScaleS (NM-PM1) wafer module

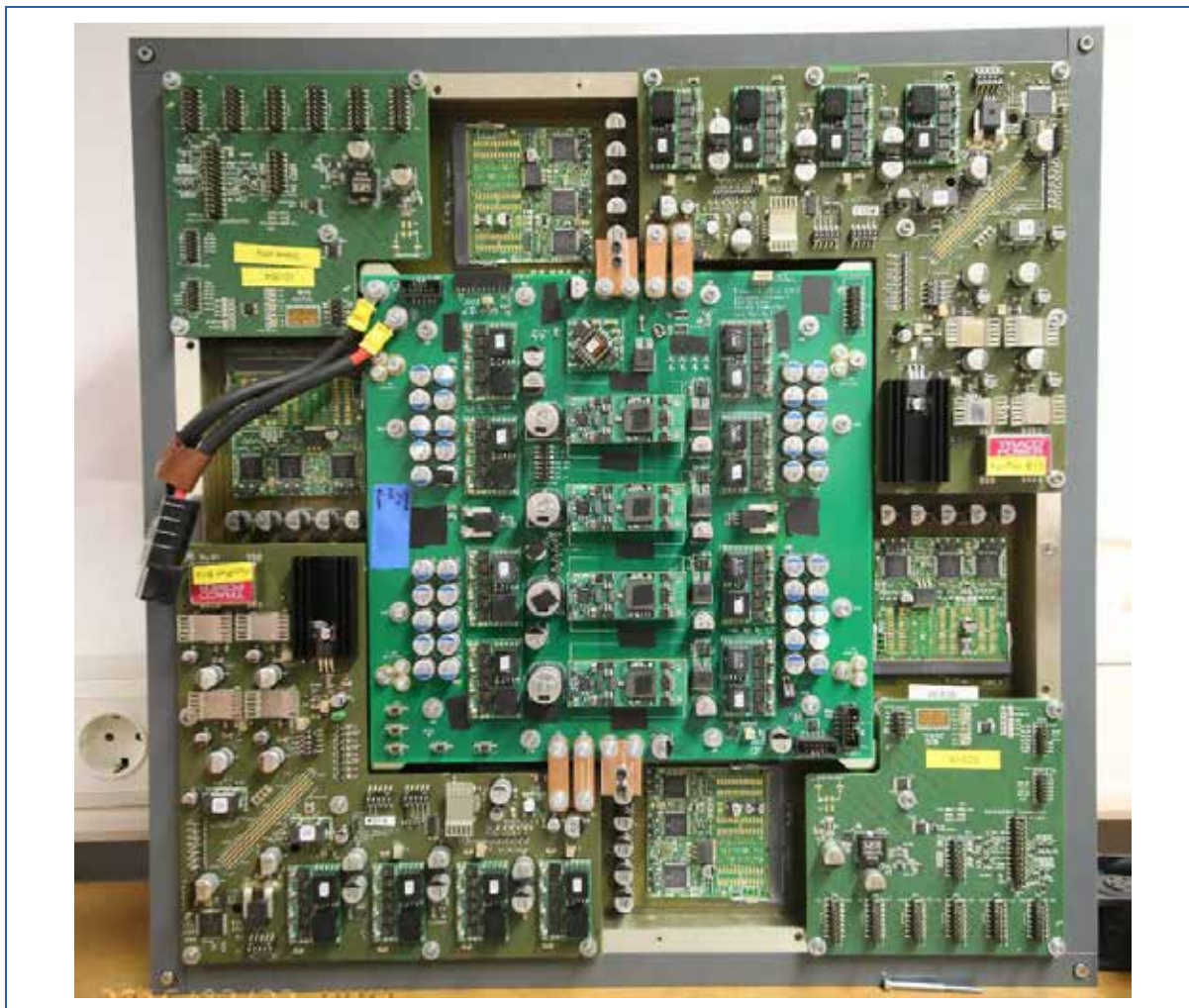


Figure 13: BrainScaleS (NM-PM1) module from the power-board side

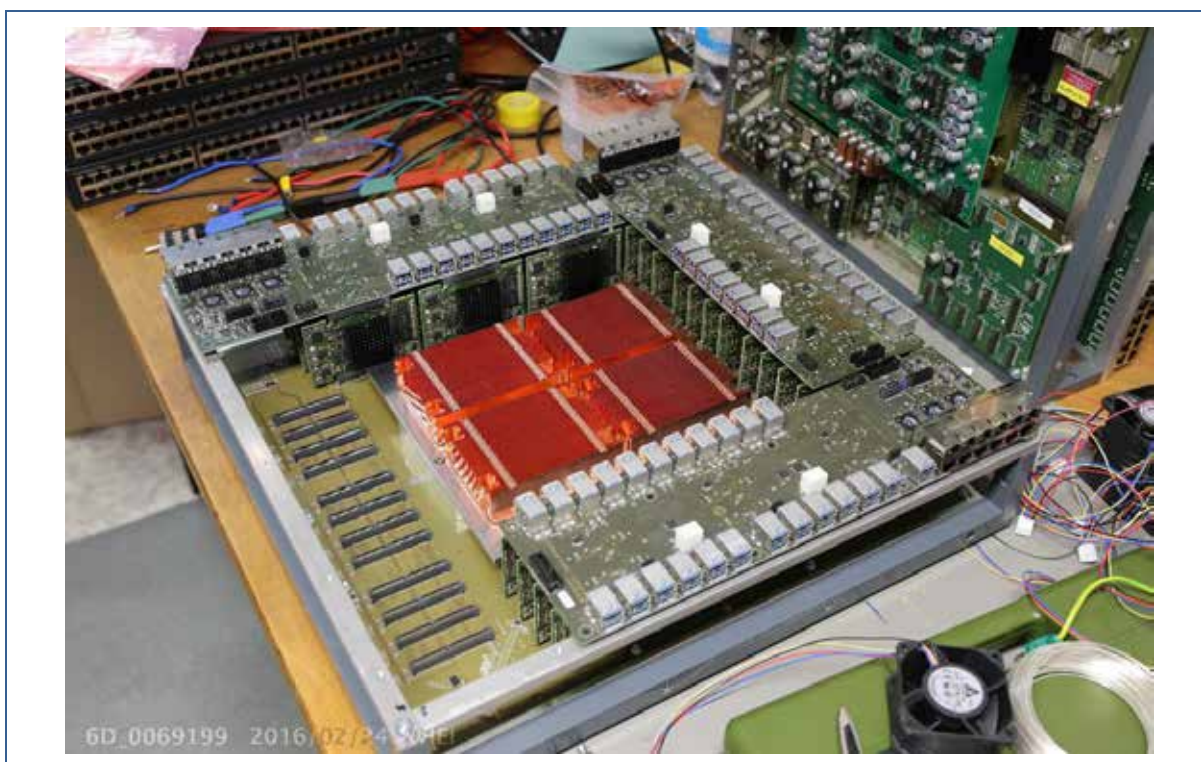


Figure 14: NM-PM1 BrainScaleS module with three of the four boards with FPGAs added

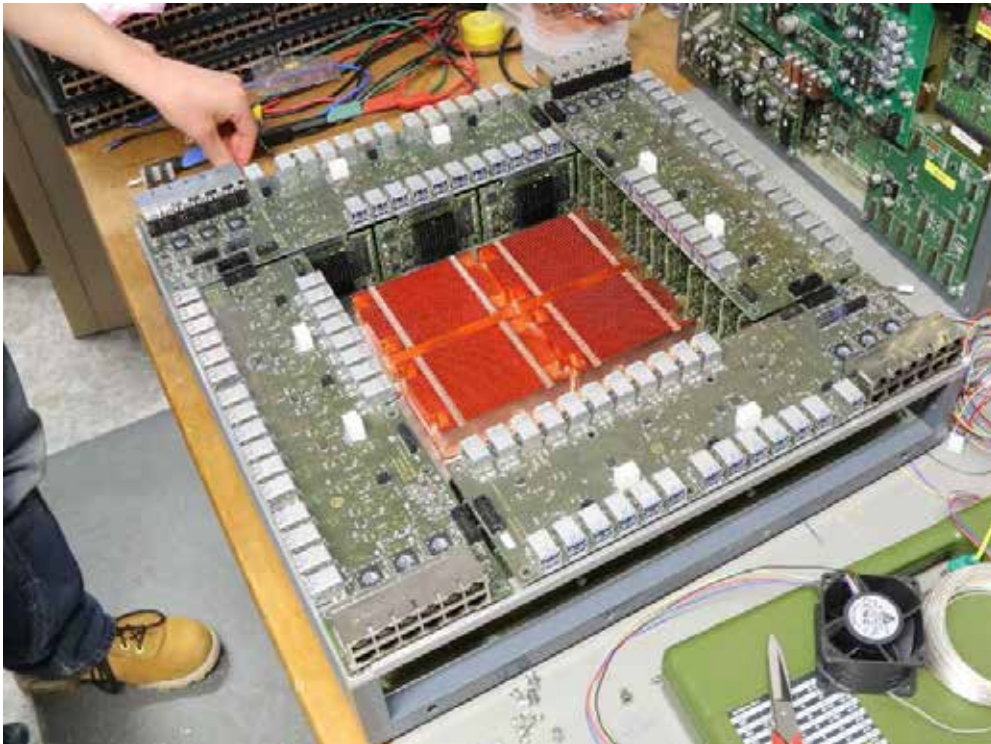


Figure 15: Fully assembled NM-PM1 BrainScaleS module, FPGA-side

Table 1: Component list for the 20 NM-PM1 BrainScaleS systems

Description	Total Number	Required
Server racks with 42 U height installed	7	7
Air conditioning system with 20 kW	2	2
Cooling drawer for the WMOD racks	20	20
Fans for wafer module cooling	240	200
Fan mounts	20	20
Elastomeric Connectors	16000	16000
Spartan6 Flyspi Boards	60	60
Post-processed HICANN V2 wafer	25	20
Post-processed HICANN V4.1 wafer	10 in prod.	bonus
Main PCB assembled	24	20
Aluminum Top Cover	24	20
Aluminum Bracket	20	20
Wafer Cooling	20	20
Control Units for Reticles (Cure PCB)	160	160

Description	Total Number	Required
Positioning Mask for Elastomeric Connectors	30	30
Main Power Supplies (PowerIt PCB)	25	20
Auxiliary Power Supply (AuxPwr PCB)	48	40
FPGA Communication PCB (FCP)	1087	960
Wafer IO Boards (hor. and vert. version)	96	80
Rack power supply 220V to 48V	6 (*3 units)	5
Raspberry Pi for system control	20	20
Intel NUC computer for ADC control	5	5
3HU crates for Sys-control components	5	5
Analog Breakout Boards (AnaB)	50	40
Adjustment assemblies	2	2
Wafer module rack fixing set (mechanics)	20	20
Analog Frontend PCB (redesign postponed, current design sufficient)	60	60
Insertion Frame	20	20
Compute/Control Nodes (standard PC cluster)	20	20
Test node	1	1
Storage Server	3	3

8. Annex B: Software and Services Included in this Platform Release

For each of the Platform's user-accessible Software Packages and Services, an entry is placed in the HBP Collaboratory's Software Catalogue:

<https://collab.humanbrainproject.eu/#/collab/19/nav/2108>

The Annex here in the Word document is just an export, created on 20 March 2016, using the <https://collab.humanbrainproject.eu/#/collab/509/nav/4818> Collab application.

Name	Title	Version	Category	Description
pyhal	PyNN Implementation for the Spikey Neuromorphic Hardware System	1e970f0	application	
vmodule	AnaRM (Analog Readout Module of HICANN Wafer Modules) Software	2108859	library	
omnibus	A pipelined hardware bus system based on OCP 2.0 written in SystemVerilog	f073266	spec	
ppu-software	Basic software libraries for the Nux processor and plasticity measurements	669c2d6	library	This repository contains basic libraries to compile software for the Nux processor. The ppu_sweep program was used for plasticity measurements in the first HICANN-DLS prototype chip.
nux	A synthesizable RISC processor implementing the Power ISA	d3384f5	spec	This repository contains the Nux Processor developed during the Brain-i-Nets, BrainScaleS, and HBP EU research projects. It is a small RISC processor implementing the Power ISA 2.06 (32 bit embedded implementation). It also has a non-standard SIMD processing unit for 8 and 16-bit fixed-point arithmetics. The design is open source and provided under the Solderpad license. For details see the LICENSE and NOTICE files.
logger	Logging Framework for UHEI Software	826c5ed	library	
spikeyhal	Low-level Interface for (USB-based) Spikey Neuromorphic Systems	260fc1e	library	Hardware Abstraction Layer of the Spikey neuromorphic system ('spikeyhal') developed by the Electronic Vision(s) Group, Kirchhoff-Institute for Physics, Ruprecht-Karls-Universität Heidelberg, Germany.

Name	Title	Version	Category	Description
brick	IC design work flow based on the waf build tool	3843b1	tool	<p>brlCk is a waf-based IC design work flow tool that offers design-independent build folder management. Multiple build runs of an IC design can be kept in parallel and can be switched between. Design source files, such as i.e. HDL code, tcl scripts and full custom design files can be copied into the components folder of the tool's directory structure. Their paths, given in a text file, will be used by brlCk to hook into and allow it to execute the necessary tasks. These tasks may include</p> <ul style="list-style-type: none"> abstract generation HDL code synthesis Place & Route Functional verification Formal verification Sign-off verification <p>Files resulting from these operations will always be avoided to be copied to the source file tree and will be put into a run-specific uniquely named result folder. In a first version, old runs will be kept, giving users the possibility to compare results of different runs (with different parameters or source code changes) to each other. However, brlCk will not allow the user to recover and/or continue these runs once he has advanced to the next run. In a future version, the user should be able to switch back to old runs and continue to work on them at any later point and at any given state. This can involve an abandonment of the strict distinction between source (i.e. input) file tree and result (i.e. output) file tree and make it necessary to copy the current task's tcl scripts and/or necessary other input files to the current run directory. To allow for continuation of past build runs, brlCk will at least have to force the user to commit the current source file tree's state to the version control system and save the commit ID for later resumption. Later on in</p>

Name	Title	Version	Category	Description
				<p>the development of brlCk, it should be possible to maintain different configurations to be able to keep one single brlCk folder, holding in it's components folder the source files of different IC designs and making it even possible to switch between different designs.</p> <p>brlCk stands for b ackend r apid IC k it. It is intended as a hardware development counterpart to symwaf2ic, the central software build flow of the Electronic Vision(s) group which is also based on waf.</p>
Neuromorphic-Computing-Platform-Job-Manager	Neuromorphic Computing Platform Job Manager	2	application	Collaboratory app for submitting jobs to the Neuromorphic Computing Platform, and retrieving/reviewing the results.
Neuromorphic-Computing-Platform-Job-Queue-Service	Neuromorphic Computing Platform Job Queue Service	1	service	Web service for submitting and retrieving simulation jobs to/from the HBP Neuromorphic Computing Platform.
Neuromorphic Platform Python client	Neuromorphic Platform Python client	0.3.0	library	Client software for the Human Brain Project Neuromorphic Computing Platform.
PyNN	PyNN	0.8.0	library	A Python package for simulator-independent specification of neuronal network models.
sPyNNaker	SpiNNaker PyNN Software	2016.001		

9. Annex C: Summary - Platform Use Case Status

Pasted in below are the latest “Technology Readiness Levels” TRL estimates for the use-cases collected on the internal wiki page https://flagship.kip.uni-heidelberg.de/w/SP9_UseCases_TRL

Status as of 29 March 2016:

9.1.1 SP9NMP-UC-001: A single run of a simple network model

Primary Actors: Bill, a computational neuroscientist

Bill has created a network model with point neurons and short-term synaptic plasticity using the PyNN API. He has simulated the model using the NEST and NEURON simulators, and now wishes to check that the results from neuromorphic hardware are comparable. Precondition: The model and experiment description are in a single Python script on Bill’s laptop.

9.1.1.1 Success Scenario. Overall TRL estimate: 5

- In a web browser, Bill navigates to the home page for the Neuromorphic Computing Platform and logs in to his user page.
- Bill can see a list of previous jobs he has run on the Platform.
- Bill clicks a button to request a new job.
- Bill copies the content of the Python script from his text editor and pastes it into the appropriate text box.
- Bill selects the Manchester system.
- Bill submits the job request.
- Bill is returned to his user page, where he can see that his new job has been added to the list of jobs with the status “in queue”.
- When the job is complete, Bill receives an e-mail containing a link to the job detail page.
- Bill clicks on the link, which opens the job detail page in his browser. This page shows that the job has successfully completed, and contains links to download the log and output data files generated by the experiment.
- Bill downloads the data files and compares the results to his NEST simulations.”

This scenario works as described.

9.1.1.2 Alternate Scenario 1 (syntax error in the script): Not currently implemented

- There is a syntax error in Bill’s script.
 - when Bill submits the job request, he is taken back to the job submission page, where a traceback of the error appears.
 - Bill corrects the error and resubmits the job.

Not currently implemented, although some components are in place. Syntax errors are currently handled as in Alternate Scenario 2.

9.1.1.3 Alternate Scenario 2 (error in the script in the output section): Overall TRL estimate: 5

- There is an error in the output data-handling section of Bill's script, after the simulation section.
 - Bill receives an e-mail informing him that the job was unsuccessful, and containing a link to the job detail page.
 - The job detail page shows the error traceback and contains a link to download the log file, enabling Bill to debug his script.

Scenario works as described.

9.1.2 SP9NMP-UC-002: A scripted run of a complex network model with input data and parameter files

Primary Actors: Carol, a computational neuroscientist Carol has developed a detailed model of a sensory system, which uses spiketiming-dependent plasticity and receives naturalistic stimulation. Even on a traditional HPC computer, the simulation takes several days to run. Carol wishes to take advantage of the large acceleration factor of the Heidelberg system to bring the run time down to a few minutes, so that she can study the effect of parameter variations. Since she expects to submit many jobs with different parameters, she wishes to script the job submission process rather than click through a website. Precondition: The model and experiment descriptions are written using the PyNN API and are in separate Python files in a public Git repository. The repository also contains parameter files, a file containing data used to construct the sensory stimuli, and a main script which reads all these files, launches the simulation and then handles the output data processing.

9.1.2.1 Success Scenario. TRL estimate: 5

- Carol downloads a Python client for the Neuromorphic Computing Platform job submission REST API.
- Using the client library, she writes a short script to submit a job to the Neuromorphic Computing Platform and retrieve the results.
- The job request script includes the name of the system (the Heidelberg system in this case), the URL of the Git repository, the path to the main script within the repository, and the list of arguments (parameter file name, etc.) required by the script.
- After submitting the job request, the script receives a URL that returns a document indicating the job status.
- The script polls the job status URL repeatedly until the job is complete, at which point the job status document contains the URLs of the output data files and the log file.
- The script downloads the output data files and saves them to the local disk.

This scenario works as described.

9.1.2.2 Alternate Scenario 1 (error in the user code). TRL estimate: 5

- There is an error somewhere in Carol's code
 - the job status document indicates there has been an error, and contains the error traceback and the URL of the log file

Works as described

9.1.2.3 Alternate Scenario 2 (public Git repository unavailable): TRL estimate: 5

- The public Git repository is unavailable
 - The job status document indicates there has been an error, and indicates the cause of the problem

Not tested, but expected to work

9.1.2.4 Alternate Scenario 3 (user script termination between job submission and completion): TRL estimate: 5

- Carol cancels the job submission script, or reboots her computer, after the job has been submitted but before the job has completed.
 - The job remains in the queue
 - When the job completes Carol receives an e-mail containing a link to the job detail page.

Works as described.

9.1.2.5 Alternate Scenario 4 (using the Python client to cancel a job): TRL estimate 4

- After submitting the job but before it has completed, Carol realizes she has made a mistake.
 - Carol uses the Python client for the Neuromorphic Computing Platform job submission REST API to cancel the job.

Implemented but not documented

9.1.3 SP9NMP-UC-003: Using the Neuromorphic Computing Platform through the Collaboratory and Brain Simulation Platform

Primary Actors: Dennis, a neuroscientist. Dennis has used the Brain Builder component of the Brain Simulation Platform to create a network model of a brain region, using point neurons. He has successfully executed a simulation of the model on the HPC Platform using the NEST simulator, and now wishes to execute the model on the Manchester hardware preparatory to beginning a collaboration with the Neurorobotics sub-project. Dennis is not comfortable with Python coding, and wishes to use the Collaboratory to perform his simulations.

Precondition: Dennis' model is available in the Collaboratory.

9.1.3.1 Success Scenario. TRL estimate: 1-2

- Dennis selects and executes a task that exports a Brain Builder model in a format suitable for execution on the Neuromorphic Platform (PyNN).
- He configures a Neuromorphic simulation job, selecting the Manchester hardware.
- He launches the job, which is then queued and executed when time is available on the hardware.
- About an hour later, Dennis receives an e-mail telling him his job has completed successfully.
- Dennis returns to the Collaboratory, from where he can access the data files generated by his simulation, as well as provenance information about the execution, e.g. what by

his simulation, as well as provenance information about the execution, e.g. what version of the hardware system was used.

Export of brain models from the Brain Simulation Platform to PyNN format could not be implemented in the Ramp-Up phase. We were over-optimistic in proposing this use case for the RU phase, and did not take into account the planned development and research schedules for the necessary inputs from SP4 and SP6. This use case is now planned for SGA1.

9.1.3.2 Alternate Scenario (a model contains features not supported by the platform): TRL estimate: 1-2

- Dennis' model contains features that are not supported by the Neuromorphic Computing Platform.
 - The export task fails, with a clear error message indicating which features are not supported.
 - Dennis consults the documentation for the Neuromorphic Computing Platform and modifies his model so that it will run on Neuromorphic Hardware.
 - He runs simulations with the modified model on the HPC Platform, and finds that the results are qualitatively unchanged.
 - He now submits a new job for the Neuromorphic Computing Platform, using the modified model, which successfully runs to completion."

9.1.4 SP9NMP-UC-004: Parameter sweeps

Primary actor: Esin, a computational neuroscientist

Description: Esin wishes to explore the parameter space of her network model. Due to its long run time, she needs to make use of the large acceleration factor of the Heidelberg system.

Preconditions: The model and experiment descriptions are written using the PyNN API in a single Python file in a public Git repository.

9.1.4.1 Success Scenario. TRL estimate: 4

- Esin writes a batch configuration file. This provides values for those parameters that will be varied across runs. She commits this to the Git repository.
- Esin downloads a Python client for the Neuromorphic Computing Platform job submission REST API.
- Using the client library, she writes a short script to submit a job to the Neuromorphic Computing Platform and retrieve the results.
- The job request script includes the name of the system (the Heidelberg system in this case), the URL of the Git repository, the path to the model script within the repository, and the path to the batch configuration file.
- After submitting the job request, the script receives a URL that returns a document indicating the job status.
- The script polls the job status URL repeatedly until the job is complete, at which point the job status document contains the URLs of the output data files and the log files from all of the runs in the batch.

- the script downloads the output data files and saves them to the local disk.

This scenario has been implemented in a modified form: in place of a batch configuration file, a Python script is used to implement the batch configuration.

9.1.4.2 Alternate Scenario (values outside valid range): TRL estimate: 3

- One of the parameter sets in the batch run contains values outside the valid range for the Neuromorphic hardware.
 - The invalid run is skipped, and a warning is written to the log file.

9.1.5 SP9NMP-UC-005: Closed-loop experiment involving a virtual environment

Primary actor: Fumiko, a roboticist.

Description: Fumiko has developed a robot simulation within a virtual environment. The robot perceives its environment via a model retina, and acts upon its environment through actuators. Communication from the retina to the robot brain model and from the brain to the actuators is via spikes. The retina, actuators and virtual environment are implemented as a C++ application.

Preconditions: Working with the developers of the Neuromorphic Computing Platform, Fumiko has successfully installed the virtual environment software on the Platform, working via remote shell access. The Python code for the brain model is in a Git repository, which has been checked out on the platform.

9.1.5.1 Success Scenario. TRL estimate: 3 (so far NM-MC only)

- Fumiko writes a Python script which connects the brain model with the retina and actuators, using a PyNN extension that connects spike-emitting and spike-receiving ports (for example, using the MUSIC interface).
- Using the REST API, Fumiko launches the job, which runs until the robot completes a pre-defined task, or until a pre-defined time limit is reached.
- When the job is complete, Fumiko receives an e-mail that contains a URL for the job status.
- Fumiko accesses this URL through the REST API and downloads the data and log files generated by the job.

A prototype of an SP10 virtual robotics environment running with SpiNNaker has been demonstrated by Felix Scheider.

9.2 Missing features / under development features

- Full provenance integration (limited provenance integration in place)
- Closed loop interfaces to SP10.

10. Annex D: Summary - Service IT Resource Planning

Product/Software Package/Service	TRL	Data Storage Capacity used by this Product	Data Storage Capacity Allocated for this Product	Location(s) of Data Storage	Data Access Protocol(s)*	Compute Resource(s) Allocated	Location(s) of Compute Resource(s) Allocated	Compute Access Protocol(s)**
SP9 user entry portal for submitting jobs to the platforms	5-6	140 MB	20 GB	Cloud (Digital Ocean datacentre in Amsterdam)	https	Currently a single webserver. Scalable according to demand.	Cloud (Digital Ocean datacentres in Frankfurt and Amsterdam) (partner CNRS)	https
SP9 NM-MC (SpiNNaker) component		Temporary storage experiment result data. for and		Local discs	(internal access)	NM-MC1 (SpiNNaker) system with 500.000 cores	Manchester (partner UMAN)	
SP9 NM-PM (BrainScaleS) component	5-6	Temporary storage experiment result data. for and	20 TB	Local discs	(internal access, except for optimized UNICORE links)	NM-PM1 (BrainScaleS) system with 20 wafers Conventional compute cluster	Heidelberg (partner UHEI)	UNICORE, SLURM, ssh
NM-PM standalone system Spikey	5-6	<i>cf. NM-PM</i>	<i>cf. NM-PM</i>	<i>cf. NM-PM</i>	<i>cf. NM-PM</i>	6 standalone Spikey systems	Heidelberg (partner UHEI)	<i>cf. NM-PM</i>

* Data Access Protocols such as GPFS, N.FS, S3, Collab storage, etc.

** Compute Access Protocols such as EC2, Task Framework, Unicore, OCCl, Slurm, ssh, gLite, Condor, etc.

11. Annex E: Summary - Service Technology Readiness Levels (TRLs) Metrics

Documentation URL - User, Developer and/or Administrator documentation is available at this URL. Strong preference should be given for publicly available documentation services.

Target User Count (TRL6+) - Target user counts (concurrent service users).

SLA Defined - The software documentation defines some Quality of Service metrics in the service documentation. These metrics may or may not be enforced by the service itself. The service has not been tested to adhere to the documented QoS metrics.

SLA Monitored - The Quality of Service metrics are monitored by a monitoring service.

SLA Enforced - The Quality of Service metrics are enforced by implementing service. If the SLA Definition indicates on 3 API/request/sec/user, there are suitable mechanisms implemented in the service to ensure these limits are not exceeded.

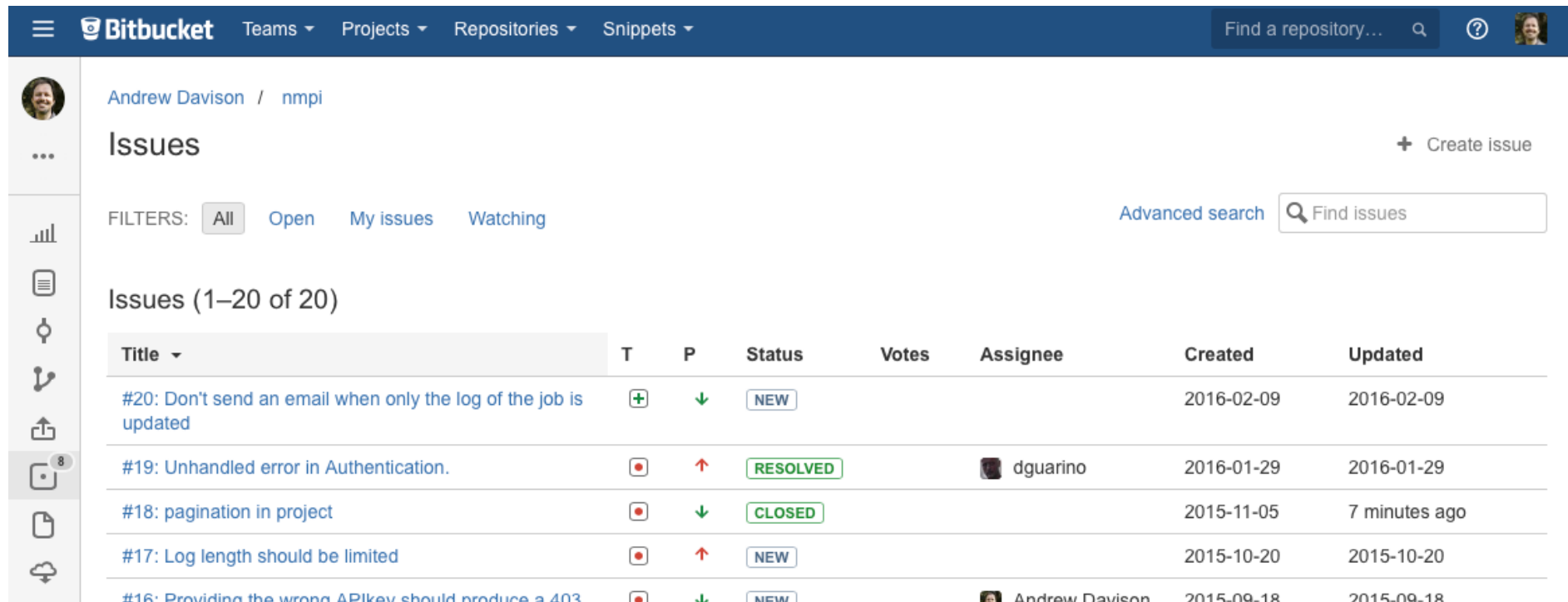
Product/Software Package/Service	Technology Readiness Level (TRL1-9)	Documentation URL	Target User Count (TRL6+)	SLA Defined (TRL7+)	SLA Monitored (TRL7+)	SLA Enforced (TRL7+)	Comments
Neuromorphic Compute Platform v1	Approaching 5 (Prototype integration - Validation of integrated system in a real-world environment = public platform release)	See also Deliverable D 9.7.5	-	-	-	-	The Neuromorphic Computer Workshop on 22 March 2016 and the HBP wide public platform release on 30 March 2016 makes the full system size described for the ramp-up phase (500.000 cores for NM-MC SpiNNaker and 20 wafers for NM-PM BrainScaleS) available.

12. Annex F: Backlog / Bug-tracking

The platform release as of 30 March 2016 is the first public platform release (V1). Especially the calibration routines for the BrainScaleS systems will be improved. Interaction with external users is expected to uncover bugs and create plenty of feature requests. For the next hardware development steps (NM-MC2 and NM-PM2 chips) the roadmap can be found in the HBP Framework Partnership Agreement (FPA).

The means for users to report bugs/problems are described in the guidebook in the Getting-Help section at http://electronicvisions.github.io/hbp-sp9-guidebook/getting_help.html. Several bug trackers are in use to track the issues:

- For the NMPI component the internal bug-tracking and work-planning is carried out at <https://trello.com/b/fjxahmd9/hbp-neuromorphic-platform-ws> (not public). The external bug-tracker is planned to be made accessible at <https://bitbucket.org/apdavison/nmpi/issues>



The screenshot shows the Bitbucket interface for the 'nmpi' repository. The 'Issues' section is active, displaying a list of 20 issues. The issues are filtered by 'All' and sorted by 'Created' date. The table below represents the data shown in the screenshot.

Title	T	P	Status	Votes	Assignee	Created	Updated
#20: Don't send an email when only the log of the job is updated	+	↓	NEW			2016-02-09	2016-02-09
#19: Unhandled error in Authentication.	●	↑	RESOLVED		dguarino	2016-01-29	2016-01-29
#18: pagination in project	●	↓	CLOSED			2015-11-05	7 minutes ago
#17: Log length should be limited	●	↑	NEW			2015-10-20	2015-10-20
#16: Providing the wrong APIkey should produce a 403	●	↓	NEW		Andrew Davison	2015-09-18	2015-09-18

- For the SpiNNaker system (NM-MC1) the main entry point for bug tracking is at the publicly accessible URL : <https://github.com/SpiNNakerManchester/sPyNNaker/issues>
- For the BrainScaleS system (NM-PM1) the bug tracking is done using the internal <https://brainscales-r.kip.uni-heidelberg.de/issues/> redmine system.

Home My page Projects Administration Help

BSS/HBP/GitViz Repository

Search: Jump to a project:

Issues

Filters: open Add filter:

Options:

Apply Clear Save

#	Project	Tracker	Status	Priority	Subject	Assigned to	Updated	Target version
<input type="checkbox"/> 2148	HALbe	Task	New	Low	Pretty free functions "freeing" Handle::ADC(Ramote,)/Hw	Eric Müller	2016-03-18 11:24	
<input type="checkbox"/> 2147	hmf-fpga	Task	Accepted	Urgent	Implement cube control via Raspberry Pi	Matja Kleider	2016-03-18 14:38	
<input type="checkbox"/> 2146	SP9 Specification D9.7.3	Task	New	Normal	Generate HTML version		2016-03-17 15:48	
<input type="checkbox"/> 2145	Vacation (and other absence)	Absence	New	Normal	Vacation	Akos Kungl	2016-03-17 12:53	
<input type="checkbox"/> 2144	ncf-hicann	Task	New	Normal	Update leaflet map of HICANN coordinates		2016-03-09 23:59	
<input type="checkbox"/> 2143	STHAL - Stateful HAL	Task	New	Normal	provide API call for get_adc_mean	Christoph Koka	2016-03-08 16:21	
<input type="checkbox"/> 2141	calix	Task	Started	Normal	workpool should obey system resources	Christoph Koka	2016-03-03 12:40	
<input type="checkbox"/> 2140	Vacation (and other absence)	Absence	New	Normal	gguette absence	Maurice Guttler	2016-03-03 09:04	
<input type="checkbox"/> 2139	admin-computing	Bug	New	Normal	Jenkins does not terminate when inner SLURM job is "Force Terminated"	Eric Müller	2016-03-09 20:24	
<input type="checkbox"/> 2138	ncf-hicann	Bug	New	Low	Some bad ground event generators don't work in random mode at 250 MHz PLL		2016-03-03 22:11	tapeout_5
<input type="checkbox"/> 2137	admin-computing	Feature	New	Normal	SLURM support for hwdb-style hardware requests	Christian Mauch	2016-03-09 20:22	
<input type="checkbox"/> 2136	admin-computing	Feature	Accepted	Normal	Support HostAPQ-SHM file cleanup via SLURM	Eric Müller	2016-03-02 11:39	
<input type="checkbox"/> 2134	HALbe	Bug	New	Normal	Parameter order should not influence output of convert_coordinates	Christoph Koka	2016-03-01 18:23	
<input type="checkbox"/> 2133	sw-aupier	Task	Started	Normal	Voltage monitoring	Alexander Imperio	2016-03-01 12:57	
<input type="checkbox"/> 2132	calix	Task	New	Normal	Evaluate and if possible reduce memory consumption		2016-03-01 11:35	
<input type="checkbox"/> 2131	admin-computing	Feature	New	Normal	SHM-IPC files remain after job termination	Eric Müller	2016-03-01 18:16	
<input type="checkbox"/> 2130	HMF System	Task	New	Normal	Implement automatic test: ADC noise measurement		2016-03-01 10:10	
<input type="checkbox"/> 2129	calix	Task	New	Normal	Calibration exits with SystemError('Worker process died unexpectedly,')	Sebastian Schmitt	2016-03-06 10:37	
<input type="checkbox"/> 2128	Platform Opening	Task	New	Normal	turn c/970 into a standalone sthal tool	Christoph Koka	2016-02-29 13:16	
<input type="checkbox"/> 2127	admin-computing	Task	New	High	Versioned calib and software modules		2016-02-29 13:17	
<input type="checkbox"/> 2126	ncf-hicann	Task	New	Normal	Add write and read time lookup table in Fg controller	Andreas Gruebl	2016-02-29 11:31	tapeout_5
<input type="checkbox"/> 2125	Marocco	Bug	New	Normal	map, modify, run (ata injecting wafer dg) fails because of missing LookupTable		2016-02-25 10:40	
<input type="checkbox"/> 2124	STHAL - Stateful HAL	Bug	Accepted	Normal	python deepcopy fails for FGConfig	Christoph Koka	2016-02-25 12:53	
<input type="checkbox"/> 2122	symVAPIC	Task	New	Normal	Archive software state	Kai Huzmann	2016-02-24 13:03	
<input type="checkbox"/> 2120	Marocco	Task	New	Normal	allow to set only one denmen's refractory set (and the others to min)		2016-02-22 17:37	
<input type="checkbox"/> 2118	HALbe	Task	New	Normal	Read out USB supply voltage on ADC boards	Andreas	2016-02-22 15:09	

Issues

View all issues

Calendar

Gantt

13. Annex G: IPR Status, Ownership and Innovation Potential

Product/Software Package/Service	IPR Status*	Owner(s)	Non-HBP users**	Innovation Potential***
SpiNNaker (NM-MC1)		UMAN and contributors	(Platform opening on 30 March 2016)	
SpiNNaker standalone boards (4- and 48-chip systems)	Patent	Cogniscience Ltd (a UMAN spin-out company)	~ 70 SpiNNaker (mainly 4-node) boards are on loan ~ 12 48-node boards have been sold to academic research labs all around the world.	Still speculative, but there is interest in Deep Learning applications.
BrainScaleS (NM-PM1)		UHEI and contributors (e.g. TUD for FPGA components)	(Platform opening on 30 March 2016)	Innovation potential - As a rapid prototyping system for neuromorphic computing with physical models. Export of functioning circuits to custom chips - As a high-performance, high speed data processing system for spatio-temporal pattern detection - As an evaluation system for generic, spike-based computing (stochastic computing, deep learning)
Spikey standalone system			Standalone USB-Spikey systems can be accessed via the internet (via the Collaboratory) and can also obtained as physical (USB-connected) box system. Currently 3 systems are given out. Several systems are accessible via the NMPI (internet access)	Innovation potential mostly as an educational platform introducing newcomers into neuromorphic computing with physical model systems. The systems will be available for collaborators.

* IPR Status: Open Source, Copyright, Patent, Trade Secret, pre-IPR (i.e. you intend to obtain some form of IPR in the future)

** If this product/software package/service is currently being used outside HBP (e.g. donated, loaned, licensed, sold), please specify by whom.

*** Innovation Potential: Potential practical applications beyond HBP, commercial and/or non-commercial.