Q Spike tools: a generic framework for parallel batch preprocessing of extracellular neuronal signals recorded by substrate microelectrode arrays

Mufti Mahmud1,2, Rocco Pulizzi1, Eleni Vasilaki1,3 and Michele Giugliano1,3,4*

1 Theoretical Neurobiology and Neuroengineering Lab, Department of Biomedical Sciences, University of Antwerp, Wilrijk, Belgium
2 Institute of Information Technology, Jahangirnagar University, Savar, Bangladesh
3 Department of Computer Science, University of Sheffield, Sheffield, UK
4 Brain Mind Institute, Swiss Federal Institute of Technology Lausanne, Lausanne, Switzerland

Edited by:
Xi Cheng, Lieber Institute for Brain Development, USA
Reviewed by:
Michela Chiappalone, Italian Institute of Technology, Italy
Leslie Samuel Smith, University of Strinig, UK
*Correspondence:
Michele Giugliano, Theoretical Neurobiology and Neuroengineering Lab, Department of Biomedical Sciences, University of Antwerp, Universiteitsplein 1, B-2610 Wilrijk, Belgium
e-mail: michele.giugliano@uantwerpen.be

Micro-Electrode Arrays (MEAs) have emerged as a mature technique to investigate brain (dys)functions in vivo and in vitro animal models. Often referred to as “smart” Petri dishes, MEAs have demonstrated a great potential particularly for medium-throughput studies in vitro, both in academic and pharmaceutical industrial contexts. Enabling rapid comparison of ionic/pharmacological/genetic manipulations with control conditions, MEAs are employed to screen compounds by monitoring non-invasively the spontaneous and evoked neuronal electrical activity in longitudinal studies, with relatively inexpensive equipment. However, in order to acquire sufficient statistical significance, recordings last up to tens of minutes and generate large amount of raw data (e.g., 60 channels/MEA, 16 bits A/D conversion, 20 kHz sampling rate: approximately 8 GB/MEA/h uncompressed). Thus, when the experimental conditions to be tested are numerous, the availability of fast, standardized, and automated signal preprocessing becomes pivotal for any subsequent analysis and data archiving. To this aim, we developed an in-house cloud-computing system, named Q Spike Tools, where CPU-intensive operations, required for preprocessing of each recorded channel (e.g., filtering, multi-unit activity detection, spike-sorting, etc.), are decomposed and batch-queued to a multi-core architecture or to a computers cluster. With the commercial availability of new and inexpensive high-density MEAs, we believe that disseminating Q Spike Tools might facilitate its wide adoption and customization, and inspire the creation of community-supported cloud-computing facilities for MEAs users.

Keywords: substrate arrays of microelectrodes, MEAs, extracellular, batch analysis, embarrassingly parallel signal-processing, cellular electrophysiology

INTRODUCTION

Among the most challenging open questions in Systems Neuroscience, structure-function relationship has raised a renewed interest. While novel ultrastructural anatomical investigations (Briggman and Denk, 2006; Mikula et al., 2012) promise to revolutionize the field, significant new progresses in our understanding of neuronal networks physiology and in pre-clinical neurotechnological applications, have been achieved by extracellularly monitoring the electrical activity of large neuronal ensembles (Rutten, 2002; Buzsaki, 2004; Wise et al., 2004; Lebedev and Nicolelis, 2006; Nicolelis and Lebedev, 2009). Complementary to high-resolution patch-clamp microscopic access and to mesoscopic non-invasive electroencephalography and functional magnetic resonance imaging, the extracellular interfacing of neurons to artificial devices has taken a considerable leap forward (Fromherz, 2006; Vassanelli et al., 2012; Spira and Hai, 2013).

Since its early introduction, extracellular recordings have been widely adopted both in academic and industrial pharmaceutical contexts, for monitoring and evoking neuronal activity in vivo and ex vivo under a variety of scientific, technological, neuropathologic, and clinical perspectives (Berdondini et al., 2006, 2009; Giugliano et al., 2008; Kim et al., 2009; Wang et al., 2012; Gortz et al., 2013; Liu et al., 2013). In addition, recent advances in real-time computing and in micro- and nanotechnologies opened brand new possibilities (Arsiero et al., 2007; Mazzatenta et al., 2007; Jain and Muthuswamy, 2008; Chen et al., 2009; Kim et al., 2009; Fendyur et al., 2011; Hai and Spira, 2012; Tian and Lieber, 2013).

However, in terms of data collection, analysis, and interpretation, multi-site extracellular recordings pose some challenges, given the large size of the raw data files acquired and the inherent complexity behind their rapid and accurate interpretation (Buzsaki, 2004; Stevenson and Kording, 2011). From Neuroinformatics perspectives, several open-source software toolboxes have been developed and released to the community over the years, addressing those issues and ultimately aimed at making electrophysiological data handling and analysis easier, faster, interactive, and more user friendly (Egert et al., 2002; Quiroga et al., 2004; Vato et al., 2004; Bonomini et al., 2005;
Exercise for example, as a generic, easily customizable, at a scenario where no further manipulation of post-processed applied, or easily customizable to include additional analysis scripts (e.g., in MATLAB), in the context of routine compounds screen-

In this work, we addressed some of the basic requirements of substrate-integrated microelectrode arrays (MEAs) users, focusing on routine multichannel data analysis in in vitro studies, where several experimental conditions are examined and several binary raw data files are collected daily. We defined two major objectives that we consider a priority in our own laboratory: (i) increasing experimental throughput by freeing the data-acquisition computers from the burden of subsequent raw-

As sample data processing primitives for demonstration purposes, we chose a minimal set of basic operations that are performed in any multi-unit activity analysis: filtering, peak-detection, sorting, and simple spike-rate analysis. T edious and long interactive analy-
sis sessions could be then replaced by an automated procedure, and most important for us, by a more rational and efficient use of the existing shared computing resources in our labora-
tory. This was accomplished by delegating and batch queuing the preprocessing of the raw data files to an in-house multi-
core server. This is controlled and monitored remotely via a simple web browser, with no (computing) programming famil-

The MEA microelectrodes were then employed to moni-

The recording in a low-humidity, electronic-friendly, conditions at 37°C, 5% CO2. The MEA microelectrodes were then employed to moni-

   Additional hardware (Figure 1A) included an acquisition computer with a PCI analog to digital board (MC Card, 64 channels A/D, 4 DIO, 16bits Multichannel Systems, Reutlingen, Germany), as well as a temperature regulator and a stimulus iso-

   Figure 1B depicts a magnification of the inner area of a MEA

1http://www.nsgportal.org/
FIGURE 1 | Recording of neuronal activity ex vivo, by means of a commercial MEA hardware platform. Our experimental setup (A) is composed by a pre-amplification and filtering (PF) stage and by an additional signal amplifier, connected via an A/D board to a dedicated data acquisition PC, which also controls a temperature controller and electrical stimulus generation (not shown). After plating and culturing mammalian primary cortical neurons ex vivo on a MEA for several days, spontaneous electrical activity is detected and recorded at each microelectrode (B), arranged as a regular 8 by 8 layout, 200 µm spacing, and displayed in real time (C). Representative raw voltage traces from four sample microelectrodes, recorded over 20 min, are shown in (D) with increasing levels of magnification, to reveal the stereotypical pattern of spontaneous multi-unit electrical activity (i.e., 1 x 1 mm²) populated by microelectrodes, and Figure 1C displays a typical recording session where the extracellular electrical activity sensed at each microelectrode can be monitored over time as an electrical potential. Figure 1D reports representative raw (off-line band-pass filtered) sample recordings, acquired over 20 min from six sample microelectrodes.

SYSTEM ARCHITECTURE

The QSpike Tools workflow is based on a client-server architecture (Figure 2A). The server is a stand-alone (powerful) computer workstation, or it is the master-node of a computers cluster, running a standard distribution of the Linux operating system. Accordingly, the individual processor cores of the server, or the computers of the cluster are configured as distributed computing nodes, as in a high-performance computing intranet. The master node also runs a web server software, capable of launching a series of server-side operations (e.g., via the common gateway interface, CGI², as in web applications) when instructed by a client computer connected to the same intranet.

The preprocessing performed by QSpike Tools includes five steps (Figure 2B). Some of them occur and progress automatically in a sequence, while others are only initiated via user interaction with the web page hosted by the master node upon selection appropriate hyperlinks. The links provide:

1. A fast and secure SFTP³ raw data transfer, from the data acquisition setup storage hard drive to the computer(s) dedicated for data preprocessing;

2. The conversion of each multiplexed raw data file into several binary files, each containing data points from a distinct recording channel;

²http://en.wikipedia.org/wiki/Common_Gateway_Interface
³http://www.openssh.com
FIGURE 2 | System architecture and workflow description. The physical configuration of computers within network(s) is reminiscent (A) of a cloud-computing architecture. Through a web browser, the user initiates (B) the workflow by requesting secure data transfer from the acquisition computer to the server, and by starting the parallel batch processing and analysis. At the end of the analysis, the server generates automatically a portable document format (PDF) document, reporting on the execution log, the activity summary, and the result summary, which is incorporated in a compressed file archive along with the analysis result. When ready, the user concludes the session by securely downloading the compressed file archive to his/her personal computer. The flowchart (C) shows the actions required by the user and the processing steps executed by the server (cluster). The steps executed in parallel are outlined by the dashed line.

(3) The scripting-based (i.e., written in MATLAB, The Mathworks, Natick, USA), fully extensible, preprocessing sequence for each file, currently including band-pass filtering, stimulation-artifacts removal, multi-unit activity detection and elementary spike-sorting (Figure 3B);

(4) The additional scripting-based (i.e., MATLAB) visualization of the (multi)unit activity extracted by the previous steps (e.g., MEA-wide synchronous bursting rate, single-channels and MEA-wide firing rate, intra-burst instantaneous discharge probability, etc.);

(5) The automated typesetting of a PDF report from a dynamically generated and compiled LaTeX\(^4\) source file, including both textual and graphical information, extracted by the previous step and secure download of the PDF report and of all intermediate and final preprocessed files (e.g., spike timestamps, spike waveforms, spike count) to the user’s personal computer, as a compressed file archive.

Provided that certain dependencies are respected (e.g., step 4 must follow the completion of step 3, across all electrodes of the same data file), all the remaining steps (i.e., 2–5) can also be batch-queued and executed in parallel (e.g., one task per core, processor, or intranet node). A flowchart explaining the detailed processing steps is shown in Figure 2C.

The QSpike Tools workflow is in fact based on the observation that any CPU-intensive preprocessing needed can be executed in parallel, independently of any other recorded channel (Denker et al., 2010). All operations, necessary for any subsequent data analysis, can be performed in parallel across channels thus directly exploiting the advantages of embarrassingly parallel scheduling (Figure 3).

It is worth to note that, in our context, parallel processing implies performing pre-, post-processing and analyses on each recorded channel, independently. This excludes sophisticated spike detection, sorting, and analyses algorithms that are useful when employing, e.g., tetrode-like arranged MEAs, whose inter-electrode distances are much smaller than those employed here. In those circumstances, correlated information from distinct channels cannot be treated independently and must be jointly analyzed. While the series of analyses of QSpike Tools can be extended to include similar analysis algorithms, this has not been yet implemented in its current version as it implies a

\[^4\]http://www.latex-project.org
FIGURE 3 | Grid computing architecture and flowchart of the data pre-, post-processing, and analysis module. (A) When jobs arrive at a grid computing environment, the jobs are decomposed into individual jobs and allocated to each processor core or cluster for completion. In a multicomputer cluster system the head node of the cluster is responsible for job-, cluster-, resource-management and scheduling leaving the application and job execution on the individual nodes of the cluster system. In QSpike Tools the benefit of a multicore single computer cluster has been exploited where the responsibilities of the head and individual nodes are handled by the grid computing software. (B) Flowchart of the execution model with exemplary pre-, post-processing and analysis modules to demonstrate and distinguish among the operations in terms of their parallelization. Dashed sections outlines the two major sets of operations, where parallelization of the tasks takes place. The remaining tasks instead operate serially. The parallelization is achieved by the qsub command, provided by the grid-scheduling environment, distributing the batch-queued jobs to the specified cores for their parallel execution.

redesign of its principle of operation, beyond the embarrassingly parallel computing.

SYSTEM IMPLEMENTATION
The system has been tested on a multi-core personal workstation (Precision, T7500, Dell, Asse-Zellik, Belgium), equipped with two six-core Xeon processors and 24 GBytes of shared memory, running the Ubuntu 5.10.10 server operating system, the Apache 6 webserver software, and MATLAB R2012a. In addition, a basic grid-computing environment was installed and set up, using the (now outdated) Sun Grid Engine (SGE, Sun Microsystems, Santa Clara, California), or the equivalent Open Grid Scheduler/Grid Engine 7. The last implements a scheduling system for the management of distributed computing resources (i.e., individual cores, processors, and computers) and it enables the definition of one or more computing queues. Upon launching a job by a special command of the scheduler, while assigning it to a specific queue, the operating system is not anymore in charge of balancing the computing load on the entire computer architecture. Instead, that job is scheduled for execution and assigned to an individual unused node, among those reserved. As mentioned, these nodes may be the processor cores of the server, as in our case, or—transparently for the user—the cores of the processor(s) of Ethernet-connected computers (Figure 2A).

Both the client PC(s) and the workstation run the OpenSSH server software, which provides a secure file transfer protocol (SFTP) 8. With the typical size of a MEA raw data file (e.g., 60 channels/MEA, 16 bits A/D conversion, 20 kHz sampling rate: approximately 8 GB/MEA, uncompressed), we found that scripted command-line SFTP performed better than drag-and-drop over network mounted shares or graphical user interface clients. Via scripted SFTP and by employing a gigabit Ethernet

---

5http://www.ubuntu.com/download/server
6http://httpd.apache.org
7http://gridscheduler.sourceforge.net
8(Win) http://www.cygwin.com, (OS X) http://www.maclife.com/article/howtos/how_enable_ssh_your_mac
switch (Catalyst 2960S-48TS-L, Cisco, Diegem, Belgium), the MEA data transfer rate was consistently approximately 100 MBps, in our daily tests.

Figure 3A sketches the simple structure of the grid-computing environment, and of the way we benefit from it. Upon arrival of a preprocessing job request, the large multiplexed (*.mcd) binary file is decomposed into 60 individual files, containing the raw waveforms of each microelectrode. Then, the same operations (Figure 3B) are applied and repeated identically to each file, so that the original job is distributed in parallel to the allocated cores. The server performs the management task of submitting the jobs to the execution queue and of checking for a free node (60 channels over, e.g., 12 cores implies an overall load of 5 jobs/core), in a way fully transparent to the end-user. The node manager is ultimately responsible for the execution of each parallelized task, and logs (standard and error) output diagnostics as separate text files.

To favor readability and user customization, most of the data transfers, user communication, job decomposition, and scheduling were coded as Bash shell9 scripts. Signals preprocessing, analysis, and automated generation of figure plots were implemented as MATLAB scripts.

The flow chart of Figure 3B depicts the various components of the preprocessing and analyses pipelines, and indicates (i.e., by dashed line boxes) the execution streams that operate in parallel. As mentioned in the Introduction, the user initiates the execution of a series of sequential steps, where input and output folder names are first provided to the non-interactive Bash scripts, which fetch the (list of) input raw data file(s) and store output results (e.g., the PDF report, the time-stamp of peak-detected multiunit activity, the analog waveform of each detected event for subsequent offline spike-sorting). Then, the individual channels and their preprocessing are distributed simultaneously as independent jobs among the cores using the qsub command, making the analysis trivially parallelized and limited only by the number of nodes available to the queue.

The decomposition of the raw data into the individual channels is the first step in the preprocessing pipeline: it is performed by a custom code, written in C and based on the vendor data access API.

After the extraction of individual channels, further preprocessing like a causal signal filtering (Quiroga, 2009), robust peak-detection detection, elementary spike sorting (i.e., as positive or negative threshold crossings), and spike waveform storage are performed (Quiroga, 2012). For instance, filtering is based on a band-pass, zero-phase digital filter of fourth order (i.e., by filtfilt10 and ellip11 MATLAB functions, included in its Signal Processing Toolbox 12) between 400 and 3000 Hz, while peak detection is based on the evaluation of the median of the raw trace, following the sample code of Wave_clus13 (Quiroga et al., 2004), which was chosen as our golden standard.

The final result of the peak-detection is the conversion of the analog raw voltage waveforms into a time-series, for each channel: these are the time-stamps of the multiunit activity, which are stored for further analyses as simple text files. As soon as these are available for all channels, they are consolidated in a single file and ready for further MATLAB-based analysis.

The post-processing and analyses pipeline starts with loading the raw single channel data, made available by the channel extraction process, and the files saved during the preprocessing stage. Analyses such as spike-train feature extraction, firing rate estimation, the instantaneous discharge probability profile calculation (Van Pelt et al., 2004), and network-wide synchronized bursting frequency estimation (Bologna et al., 2010) are performed on the preprocessed data. Each of them produces one or more figures, as well as numerical information that are also included in the portable document format (PDF) report generated—see the Supplementary Material. Finally, the report, the figures, and all the intermediate (text and MATLAB binary) files are compressed as a file archive: for each raw data file provided initially, a single file archive is generated by the workflow and available for subsequent user’s download.

RESULTS

PERFORMANCES AND SAMPLE PREPROCESSING

The workflow discussed in the previous sections, has been employed daily in our laboratory and extensively tested. For demonstration purposes, we selected typical data files, containing the spontaneous electrical activity of ex vivo developing networks of primary cortical neurons, and we provide the PDF report automatically generated by QSpike Tools of its analysis as a Supplementary Information. By a simple web interface, as shown in Figure 4, where the various Bash scripts are linked, we experienced that several users with limited computer proficiency could perform smoothly server-side operations, such as visualizing the status of the server queues, clearing the input and output directories from previous data analysis sessions, initiating file transfer, and finally launching the data preprocessing. The same operations could be also performed remotely, form home, upon the virtual private network (VPN14) imposed by our university.

Figure 4 shows a screenshots of the web-interface of QSpike Tools. Upon navigating to the web address of the server, the user is presented with page 1, where an identification is required. This takes the user to the page 2, containing hyperlinks to most of QSpike Tools functionalities: (a) visualizing the current status of the queue, (b) checking the correct availability of the (transferred) files in the input/output directories, (c) transferring data and report to and from the server, (d) clearing the input/output directories of their content upon completion of the analysis, and (e) starting the work-flow by selecting the required number of cores to be used. Once the user opts to transfer raw data files to the server, page 3 appears with additional options. At the end, page 4 is shown with progress and diagnostic information. Finally, as the user wants to download the report, page 5 appears and enables downloading the corresponding file to the user’s PC.

9http://en.wikipedia.org/wiki/Bash_(Unix_shell)
10http://www.mathworks.com/help/signal/ref/filtfilt.html
11http://www.mathworks.com/help/signal/ref/ellip.html
12http://www.mathworks.nl/products/signal/
13http://www2.le.ac.uk/centres/cstn/wave-clus-docs/
14http://en.wikipedia.org/wiki/Virtual_private_network
Mahmud et al. QSpike tools

FIGURE 4 | Screenshots of the web-interface. Each window is numbered to denote a separate stage of the workflow, and consist in: (1) the user-identification; the (2) main control webpage; the (3) file transfer interface from the data acquisition PC to the master node; (4) the result of the preprocessing; and (5) the file transfer from the master node to the user PC.

Individual letter labels in (2) represent grouped functionalities, such as the visualization (a) of the status of the computing queues, the availability check (b) for the data files in the input and output directories, the file transfer and management functions (c, d), and finally (e) the initiation of the parallelized preprocessing and analysis, with an option to select the destination queue.

For the sake of testing and comparison, we configured and run QSpike Tools on sample binary (*.mcd) data files of two different sizes (i.e., approximately 1.5 GB for 8 min and approximately 3.5 GB for 20 min recording). As the execution times depend on both the raw data file and the number of multunit events, for the sake of fair comparison we executed repeatedly QSpike Tool analysis for 10 times over the very same files. We specifically selected two files, from each file size groups.

We employed four distinct predefined queues (i.e., with 1, 4, 8, and 12 reserved cores) to compare the User Execution Times\(^{15}\) (Figure 5). Confirming the embarrassingly parallelization of the task, we found that execution time reduced significantly ($p < 0.05$, ANOVA; sublinearly) with an increasing number of cores available (see Figure 5A), with maximum and minimum execution times ranging from $34.7 \text{ min} \pm 10 \text{ s}$ to $10.8 \text{ mins} \pm 17 \text{ s}$ or from $31.5 \text{ min} \pm 5 \text{ s}$ to $4.3 \text{ min} \pm 6 \text{ s}$ for large and small files, respectively.

Despite input files were identical, their repeated analysis led to variable execution times. In order to trace the sources of such variability and provide a preliminary profiler analysis of QSpike Tools, we considered separately the steps performed during the entire workflow, launching manually three subprocesses: (i) raw data channel demultiplexing, (ii) pre-processing of analog voltages (Figure 3B), and (iii) post-processing (Figure 3B). We then monitored, by standard Linux system calls, the occurrence of three computationally intensive operations managed by the operating systems: voluntary context-switching\(^{16}\) (VCS), minor page faults\(^{17}\) (Minor PF), and major page faults (Major PF).

\(^{15}\)http://en.wikipedia.org/wiki/Time_(Unix)

\(^{16}\)http://en.wikipedia.org/wiki/Context_switch

\(^{17}\)http://en.wikipedia.org/wiki/Page_fault
FIGURE 5 | Execution times and efficiency, for an increasing numbers of cores reserved. Box plots (A) with box height showing 25–75% of the sample values were used to represent maximum and minimum (whiskers), median ("−") and mean ("+") execution times, respectively, which were all significantly different (\( p < 0.05 \)) based on the number of cores used. The vertical line in the middle separates the data representation corresponding to distinct file sizes (3.5 vs. 1.5 GB). Pearson’s linear correlation coefficients (B,C) show that the execution times of individual sub-processes are correlated to, certain computationally intensive operations performed by the operating system such as, minor page faults (Minor PF), and voluntary context-switching (VCS) (\( \ast \ast \ast \ p < 0.0001; \ast \ p < 0.05 \)) in case of large file size (B). Though major page faults (Major PF) were noticed while analyzing the large file, they had either negative or no correlation to the execution times. The execution times of the first two sub-processes for the small file were mainly correlated to Minor PF (\( \ast \ast \ast \ p < 0.0001; \ast \ p < 0.01 \)) (C). Overall, negligible amount of Major PF occurred during the execution of the small file and only when large number of cores was used, correlation between execution times and VCS were noticed. The bars in (B,C) are plotted using the same color code of (A). The efficiency of parallelization was also quantified (D) as referred to the slowest execution time when a single core was used: continuous lines are best-fit logarithmic plots, whose mean squares were 0.8807 and 0.9306 for 3.5 and 1.5 GB file sizes, respectively. The mean execution times (E) for both file sizes also show the reduction of the execution time, for an increasing number of cores available.

We found that the execution times are significantly correlated to the occurrence of Minor PF and of VCS, in case of the large input file (Figure 5B) (\( p < 0.0001 \) and \( p < 0.05 \), respectively). The same occurs for smaller input files, particularly during channel extraction and pre-processing sub-processes (Figure 5C).

We then noticed that the execution time with the highest number of cores was found to be more sensitive to page faults and context-switching. This may be explained in terms of the fixed amount of physical memory, as its allocation per core decreases with increasing number of cores. As a consequence, memory
FIGURE 6 | Average voltage waveforms, corresponding to (positive or negative) peak-detection events at each microelectrode. Each subplot represents graphically the $8 \times 8$ layout of the MEAs employed in this study (see the Methods). The average voltage trajectories (thick black lines) of the positive (A) or negative (B) threshold crossing events are displayed for each microelectrode, together with its point by point variability (gray shading, i.e., standard deviation), and with the number of events among brackets. The numbers on the right hand side of each subplot denote the labeled electrode name.
FIGURE 7 | Experimental outcome at a glance. The distribution of the number of electrodes (A) that detected activity with significant occurrence frequency is displayed, together with the MEA-wide distribution (B) of the interspike intervals that reveal a bimodal profile, corresponding to the recurring transient, MEA-wide synchronization of the electrical activity.

demanding jobs required to run in parallel will have less allotted memory and result in frequent page faults and context-switches (Tay and Zou, 2006). Based on the necessity one may try to reduce the occurrence of page faults by implementing available methods in the literature for better memory management (Zhou et al., 2004).

The efficiency $E$ across distinct queue size, was calculated with respect to the execution times required by a single-core system as $E = \frac{T_{\text{max}}}{\Delta T} \times 100\%$, with $T_{\text{max}}$ the execution time of the slowest execution—referred to a single core. As expected, the execution efficiency increases (Figure 5D) and the mean execution time maintains a decaying trend (Figure 5E) with increasing number of cores used.

We have excluded those steps when computing the execution times, such as file transfer to and from the master node file system, since these are dependent on physical characteristics of the Ethernet network as well as of user interactions. The significant differences in the execution times indicate that using more powerful computers with significantly large number of cores is advantageous for a large set of raw data files. We note however that suboptimal memory management by MATLAB parallel instances still deserve attention, as the proportional decrease in the average execution time for large data files differed from the execution time for smaller files.

For the sake of illustration, we further comment and discuss briefly some of the standard analyses performed by QSpikes Tools. The first step in the analysis pipeline is to display graphically the waveforms detected at each electrode by the peak-detection algorithm. This allows an elementary spike sorting procedure, discriminating between positive- or negative-threshold crossing. It also enables the user to perform a quantification of the average voltage data trajectory amplitude, its confidence interval, as well as the overall number of events detected. Besides serving as a quality assessment of the raw signals recorded and of the viability of the culture examined (Figure 6), next to each subplot the indication of the total number of multiunit events detected at every electrode provides immediate feedback on the significance of the average waveform displayed therein. This is also particularly useful when distinct microelectrode materials are used (e.g., carbon nanotubes or nano-crystalline diamonds coated electrodes), and when the majority of events detected by a given electrode are monophasic, biphasic, or triphasic extracellular action potentials.

Complementary information is also displayed in the form of a histogram (Figure 7A), which quantifies the number of micro-electrodes that detected a sufficiently large number of events, higher than a 0.02 Hz minimal occurrence frequency. Finally, the interspike interval (ISI) distribution is also provided across the entire MEAs, merging together all the events (Figure 7B), to ultimately reveal whether or not a bimodal distribution is present and corresponds to a mature bursting electrical phenotype of the culture.

Finally, a graphical display of the first few minutes of each recording is also produced, as a raster plot of the individual spike times across channels (Figure 8). In our own experience, this provides a rather immediate overview of the recording session and on the viability of the culture. Based on a simple eyeball estimation of the occurrence of MEA-wide synchronized events, and from a detail of the largest synchronized event at increasing temporal resolutions (Figure 9), a non-expert user can gain immediate insight on whether the typical expected electrical patterned activity occurred and the extent of the most notable synchronous transient, useful when very small recordings are performed.

The pdf-report, generated automatically at the end of the pre-processing, is provided as Supplementary Information, and offers diagnostic details on the execution of the workflow (i.e., the queue name, start and end time of the processes, and total execution time), as well as of the quantitative information about the data (i.e., total recording duration, total number of samples, sampling
The multiunit spontaneous activity is displayed as raster-plot (A) across the detecting microelectrodes for the first 5 min of each data file, and its corresponding spike count is computed (B) to reveal the MEA-wise stereotypical episodic synchronization of neuronal activity.

**DISCUSSION AND CONCLUSION**

Starting in the early 2000s, the MEA-Tools (Egert et al., 2002) laid the foundation for user-friendly analysis of MEA data, and eventually became a platform-independent, open source framework for the analysis of neuronal activity data, called FIND (Meier et al., 2008). The toolbox provided for the first time the community with a convenient graphical user interface, and with a set of MATLAB routines, for accessing, visualizing, and analyzing MEAs data. Its minor shortcoming of being centered to a specific hardware system was finally overcome by the DATA-MEAns (Bonomini et al., 2005), which operates and produces ASCII files to be used by other graphical, mathematical or statistical packages. The Neural Signal Manager package (Novellino et al., 2009) was then designed to perform sophisticated analysis of spike and bursting activities, and the SPYCODE package (Bologna et al., 2010) increased the repertoire of standard and advanced tools including cross-correlation analysis and neuronal avalanche detection. MEABench (Wagenaar et al., 2005), on the

rate, number of active electrodes, number of spikes, number of bursts, mean burst duration, standard deviation of burst duration, mean and standard deviation of the inter-burst-intervals, burst detection threshold, etc.).
other hand, was designed to provide advanced means at real-time control of the data acquisition hardware, removal of stimulation artifact, detection of spikes, visualization of voltage traces with spike raster plots.

Instead of investing on our own featured package or toolbox, we explicitly focused on the most time-consuming aspect of preprocessing large MEA data files. We do understand that MATLAB is an interpreted language and is not the fastest possible solution to perform stereotyped operations (e.g., filtering and peak-detection). Nonetheless, it has inherent advantages that we strongly favored it as many (new) algorithms are very often proposed, shared, and validated by the community as MATLAB code, making their rapid prototyping or implementation easier. We aimed at taking advantage of those existing analysis tools, but only handling much smaller and portable preprocessed files and we obtained a significant overall reduction in the analysis. For some applications (e.g., a pharma industrial context), however a minimal set of basic analyses and their automated reporting as a PDF file, as we demonstrated here, could instead be sufficient to increase user access and throughput of MEA analysis.

QSpike Tools should be then considered as complementary to existing tools, and its advantages make it suitable for performing automated preprocessing of large datasets, prior to any user interactive (advanced) analysis session.

The limitation of the current version of QSpike Tools is the compatibility with a single proprietary input file format (i.e., *.mcd), as generated by the acquisition software of the commercial platform we use (i.e., Multichannel Systems). Overcoming this limitation is a task for the future and it will be rather simple, in all the cases of raw data formats for which a file interpreter is already available for MATLAB, under Linux. Along these lines, we will also rely on community-supported vendor-neutral initiatives, such as the Neuroshare API library of functions (http://neuroshare.sourceforge.net). In addition, we also aim at (i) enriching the user interface of QSpike Tools, (ii) including a user-specific configuration file to enable custom sets of analyses, and (iii) extending QSpike Tools to in vivo data.

Along the lines of the creation of community-supported Neuroinformatics shared facilities for numerical simulations, we launch the proposal for one or more facilities dedicated to automated MEA data analysis. Finally, our work will be made available through the International Neuroinformatics Coordinating Facility, INCF (http://incf.org/) website as well as at https://sites.google.com/site/qspiketool for the community.

INFORMATION SHARING STATEMENT

QSpike Tools and its installation manual are available on request from https://sites.google.com/site/qspiketool.

AUTHORS AND CONTRIBUTORS

This work was carried out in close collaboration between all co-authors. Eleni Vasilaki and Michele Giugliano first defined the research theme and contributed an early software architecture. Mufti Mahmud, Rocco Pulizzi, and Michele Giugliano further implemented and refined methods and algorithms, carried out the data analysis, and preprocessing routines design. Mufti Mahmud and Michele Giugliano wrote the paper. All authors have contributed to, seen and approved the final manuscript.

ACKNOWLEDGMENTS

We are grateful to Dr. G. Musumeci for sharing test electrophysiological recordings, to Mr. D. Van Dyck and Mr. M. Wijnants for excellent technical assistance, and to Dr. L. Gambazzi and Mr. R. A. Barone for prior programming assistance. Financial support from the 7th Framework Programme of the European Commission (Marie Curie Networks “NAMASEN” and “NEUROACT,” contracts no. 264872 and 286403; and Future Emerging Technology project “ENLIGHTENMENT” contract no. 284801), the Research Foundation Flanders (grant no. G.0888.12N), and the Interuniversity Attraction Poles Program (IUAP), initiated by the Belgian Science Policy Office, is kindly acknowledged. Access to the supercomputing facilities CalcUA (University of Antwerp) is also acknowledged.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: http://www.frontiersin.org/journal/10.3389/fninf.2014.00026/abstract

REFERENCES

Abdoun, O., Joucla, S., Mazzocco, C., and Yvert, B. (2011). NeuroMap: a spline-based interactive open-source software for spatiotemporal mapping of 2D and 3D MEA data. Front. Neuroinform. 4:119. doi: 10.3389/fninf.2010.00119

Arsiero, M., Luscher, H. R., and Giugliano, M. (2007). Real-time closed-loop electrophysiology: towards new frontiers in in vitro investigations in the neurosciences. Arch. Ital. Biol. 145, 193–209.

Berdondini, L., Chippalone, M., van der Wal, P. D., Imfeld, K., de Rooij, N. F., Koudelka-Hep, M., et al. (2006). A microelectrode array (MEA) integrated with clustering structures for investigating in vitro neurodynamics in confined interconnected sub-populations of neurons. Sens. Actuators B Chem. 114, 530–540. doi: 10.1016/j.snb.2005.04.042

Berdondini, L., Massobrio, P., Chiappalone, M., Tedesco, M., Imfeld, K., Maccione, A., et al. (2009). Extracellular recordings from locally dense microelectrode arrays coupled to dissociated cortical cultures. J. Neurosci. Methods 177, 386–396. doi: 10.1016/j.jneumeth.2008.10.032

Bologna, L. L., Pasquale, V., Garofalo, M., Gandolfo, M., Baljon, P. L., Maccione, A., et al. (2010). Investigating neuronal activity by SPYCODE multi-channel data analyzer. Neural Netw. 23, 685–697. doi: 10.1016/j.neunet.2010.05.002

Bonomini, M. P., Ferrandez, J. M., Bolea, J. A., and Fernandez, E. (2005). DATA-MEANs: an open source tool for the classification and management of neuronal ensemble recordings. J. Neurosci. Methods 148, 157–164. doi: 10.1016/j.jneumeth.2005.04.008

Brigman, K. L., and Denk, W. (2006). Towards neural circuit reconstruction with volume electron microscopy techniques. Curr. Opin. Neurobiol. 16, 562–570. doi: 10.1016/j.conb.2006.08.010

Buzsaki, G. (2004). Large-scale recording of neuronal ensembles. Nat. Neurosci. 7, 446–451. doi: 10.1038/nn1233

Chen, D., Li, X., Cui, D., and Wang, L. (2013a). GPGPU-enabled synchronization measurement of multiple brain regions upon nonlinear interdependence analysis. IEEE Trans. Neural Syst. Rehabil. Eng. 21, 33–43. doi: 10.1109/TNSRE.2013.2258939

Chen, D., Lu, D. C., Tian, M. W., He, S., Wang, S. T., Tian, J., et al. (2013b). Towards energy-efficient parallel analysis of neural signals. Cluster Comput. 16, 39–53. doi: 10.1007/s10586-011-0175-6

Chen, D., Wang, L. Z., Ouyang, G. X., and Li, X. L. (2011). Massively parallel neural signal processing on a many-core platform. Comput. Sci. Eng. 13, 42–51. doi: 10.1109/MCSE.2011.20
Chen, Y. Y., Lai, H. Y., Lin, S. H., Cho, C. W., Chao, W. H., Liao, C. H., et al. (2009). Design and fabrication of a polyimide-based microelectrode array: application in neural recording and repeatable electrolytic lesion in rat brain. J. Neurosci. Methods 182, 6–16. doi: 10.1016/j.jneumeth.2009.05.010

Cui, Y., Yu, L., Brorsen, S. L., Ding, M., and Liang, H. (2008). BSMART: a Matlab/C toolbox for analysis of multichannel neural time series. Neural Netw. 21, 1094–1104. doi: 10.1016/j.neunet.2008.05.007

Denker, M., Wiebelt, B., Fliegner, D., Diesmann, M., and Morrison, A. (2010). “Practically trivial parallel data processing in a neuroscience laboratory,” in Analysis of Parallel Spike Trains, Vol. 7, eds S. Grün and S. Rotter (New York, NY: Springer US), 413–436. doi: 10.1007/978-1-4419-5675-0_20

Egert, U., Knott, T., Schwarz, C., Naworet, M., Brandl, A., Rotter, S., et al. (2002). MEA-Tools: an open source toolbox for the analysis of multi-electrode data with MATLAB. J. Neurosci. Methods 117, 33–42. doi: 10.1016/S1096-0707(02)00045-6

Fendyur, A., Mazurski, N., Shappir, J., and Spira, M. E. (2011). Formation of essential ultrastructural interface between cultured hippocampal cells and gold mushroom-shaped MEA towards “IN-CELL” recordings from vertebrate neurons. Front. Neuroeng. 4:14. doi: 10.3389/fneng.2011.00014

Fromherz, P. (2006). Three levels of neuroelectronic interfacing. Ann. N.Y. Acad. Sci. 1093, 143–160. doi: 10.1196/annals.1382.011

Giugliano, M., La Camera, G., Fusi, S., and Senn, W. (2008). The response to cortical neurons to in vivo-like input current: theory and experiment: II. Time-varying and spatially distributed inputs. Biol. Cybern. 99, 303–318. doi: 10.1007/s00422-008-0279-0

Gortz, P., Siebler, M., Ihl, R., Henning, U., Luckhaus, C., Supprian, T., et al. (2013). Multielectrode array analysis of cerebrospinal fluid in Alzheimer’s disease versus mild cognitive impairment: a potential diagnostic and treatment biomarker. Biochim. Biophys. Acta. 1839, 295–301. doi: 10.1016/j.bbrev.2013.09.009

Huang, Y., Li, X., Li, Y., Xu, Q., Lu, Q., and Liu, Q. (2008). An integrative analysis platform for multiple neural spike train data. J. Neurosci. Methods 172, 303–311. doi: 10.1016/j.jneumeth.2008.04.026

Hwang, W. J., Lee, W. H., Lin, S. J., and Lai, S. Y. (2013). Efficient architecture for spike sorting in reconfigurable hardware. Sensors 13, 14860–14887. doi: 10.3939/sj11114860

Jain, T., and Muthuswamy, J. (2008). Micro electrode array (MEA) platform for targeted neuronal transfection and recording. IEEE Trans. Biomed. Eng. 55, 827–832. doi: 10.1109/TBME.2007.914403

Just, T., Stoos, S., Klefenz, E., and Husar, P. (2013). Spike sorting algorithm for multichannel MEA recordings based on cross correlation. Biomed. Tech. 58(Suppl.1). doi: 10.1515/bmt-2013-1198

Kim, E. T., Kim, C., Lee, S. W., Seo, I. M., Chung, H., and Kim, S. J. (2009). Feasibility of microelectrode array (MEA) based on silicone-polyimide hybrid for retina prosthesis. J. Neurosci. Methods 177, 194–198. doi: 10.1016/j.jneumeth.2008.09.033

Quiroga, R. (2009). What is the real shape of extracellular spikes? J. Neurosci. Methods 177, 194–198. doi: 10.1016/j.jneumeth.2008.09.033

Quian Quiroga, R. (2012). Spike sorting. Curr. Biol. 22, R45–R46. doi: 10.1016/j.cub.2011.11.005

Quiroga, R. Q., Nadasdy, Z., and Ben-Shaul, Y. (2004). Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. Neural Comput. 16, 1661–1687. doi: 10.1162/08997660476521631

Rutten, W. L. C. (2002). Selective electrical interfaces with the nervous system. Ann. Rev. Biomed. Eng. 4, 407–452. doi: 10.1146/annurev.bioeng.4.020702.134327

Warland, S. B. (2004). Cortical neural prosthetics. Ann. Rev. Neurosci. 27, 487–507. doi: 10.1146/annurev.neuro.27.070203.144233

Spira, M. E., and Hai, A. (2013). Multi-electrode array technologies for neurosience and cardiology. Nat. Nanotechnol. 8, 83–94. doi: 10.1038/nnano.2012.265

Stevenson, I. H., and Kording, K. P. (2011). How advances in neural recording affect data analysis. Nat. Neurosci. 14, 139–142. doi: 10.1038/nn.2737

Tay, Y. C., and Zou, M. (2006). A page fault equation for modeling the effect of memory size. Perform. Eval. 63, 99–130. doi: 10.1016/j.peva.2005.01.007

Tian, B. Z., and Liu, C. M. (2013). Synthetic nanoelectronic probes for biological cells and tissues. Ann. Rev. Anal. Chem. 6, 31–51. doi: 10.1146/annurev-anchem-062012-092623

Van Pelt, J., Wolters, P. S., Corner, M. A., Rutten, W. L. C., and Ramakers, G. J. A. (2004). Long-term characterization of firing dynamics of spontaneous bursts in cultured neural networks. IEEE Trans. Biomed. Eng. 51, 2051–2062. doi: 10.1109/TBME.2004.827936

Vassanelli, S., Mahmud, M., Girard, S., and Maschietto, M. (2012). On the way to in vivo modelled framework for neural data analysis. J. Neurosci. Methods 21, 1085–1093. doi: 10.1016/j.jneumeth.2008.06.019

Vassanelli, S., Mahmud, M., and Vassanelli, S. (2009). FEPrWAVELAB-a toolbox for multi-channel analysis of time-frequency transformed event related potentials. J. Neurosci. Methods 161, 361–368. doi: 10.1016/j.jneumeth.2006.11.008

Wilson, J. A., and Williams, J. C. (2009). Massively parallel signal processing using the graphics processing unit for real-time brain-computer interface feature extraction. Front. Neuroeng. 2:11. doi: 10.3389/neuro.16.011.2009

Wise, K. D., Anderson, D. J., Harki, J. F., Kipke, D. R., and Najafi, K. (2004). Wireless implantable microsystems: high-density electronic interfaces to the nervous system. Proc. IEEE 92, 76–97. doi: 10.1109/PROC.2003.820544

Ye, B., Mak, T., Li, X., Smith, L., Sun, Y., and Poon, C. S. (2012). Stream-based Hebbian eigenfilter for real-time neuronal spike discrimination. Biomed. Eng. Online 11:18. doi: 10.1186/1475-925X-11-18
Zhou, P., Pandey, V., Sundaresan, J., Raghuraman, A., Zhou, Y., and Kumar, S. (2004). "Dynamic tracking of page miss ratio curve for memory management,” in Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems (Boston, MA: ACM), 177–188.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Received: 12 December 2013; accepted: 27 February 2014; published online: 19 March 2014.

Citation: Mahmud M, Pulizzi R, Vasilaki E and Giugliano M (2014) QSpike tools: a generic framework for parallel batch preprocessing of extracellular neuronal signals recorded by substrate microelectrode arrays. Front. Neuroinform. 8:26. doi: 10.3389/ fninf.2014.00026

This article was submitted to the journal Frontiers in Neuroinformatics. Copyright © 2014 Mahmud, Pulizzi, Vasilaki and Giugliano. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) or licensor are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.