Riccardo Volonterio

Web-based Human- and Machine-Driven computation

Tesi di laurea specialistica

Relatore: Alessandro Bozzon
Co-relatore: Luca Galli

Politecnico di Milano
Polo regionale di Como
Dipartimento di Elettronica e Informazione
Settembre 2012

La citazione è un utile sostituto dell'arguzia.

— Oscar Wilde

Dedicato a tutti gli appassionati di LaTeX.

CONTENTS

1 THE BACKGROUND 1
1.1 Crowd-based computation distribution 1
1.1.1 Human computation 1
1.1.2 Parasitic computing 2
1.2 Enabling web-based distributed computation 4
1.2.1 HTML5 4
1.2.2 WebCL 7

2 THE MODEL 9
2.1 Computation model 9
2.2 Task distribution model 10
2.3 Task and performer model 10
2.4 CrowdSearcher???? 10

3 THE USE-CASES 11
3.1 Human-driven 11
3.2 Machine-driven 11
3.3 Human and machine driven (Hybrid) 11

4 IMPLEMENTATION AND EVALUATION 13
4.1 Architecture 13
4.2 Performance comparison??? 13

A CONCLUSION AND FUTURE WORKS 15

LIST OF FIGURES

LIST OF TABLES

SOMMARIO

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

ABSTRACT

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Abbiamo visto che la programmazione è un'arte,
perché richiede conoscenza, applicazione, abilità e ingegno,
ma soprattutto per la bellezza degli oggetti che produce.

— Donald Ervin Knuth

RINGRAZIAMENTI

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Como, Settembre 2012 L. P.

INTRODUCTION

Web users are every day more interested in solutions for difficult multi-domain problems. Search engines are trying to satisfy this need analyzing semantic data from the web. Questions like "*What is the best restaurant in Berlin?*" or "*What is a cheap accommodation near the Colosseum in Rome?*" are common tasks, to which people all over the world daily try to find a solution.

In this scenario has emerged the groundbreaking paradigm of Crowdsourcing: internet connects millions of people sitting at home at their computers; why don't combine their intelligences and coordinate them to accomplish specific tasks? Wikipedia is a good example of crowdsourcing: a free encyclopedia that anyone can edit. Research work has addressed crowdsourcing from different perspectives and within various communities, including information retrieval, databases, artificial intelligence, social science. Researchers have investigated several areas in which crowdsourcing paradigm could be applied and several methods to engage this *hive* of interconnected intelligences to a common goal.

A recent work from Bozzon *et al.* [2012] proposed a characterization of crowdsourcing toward information retrieval tasks, i.e. activities where the goal is to find informations about a specific topic. The proposed approach, named "CrowdSearcher", bases on the following observations: during a search process people often make up their minds by combining results from search engines, investigations on vertical portals, and opinions gathered within their friends and trusted people circles. CrowdSearcher proposed a framework to "crowdsource" this activities to social networks websites, where crowdsearching processes, are characterized by:

1. the interaction with performer recruited on social networks, so to assign search-oriented, human computation tasks;

2. the characterization of users over multiple social networks, so to holistically catch their actual domains of expertise;

3. the need for a broad set of addressable expertise, as crowdsearch queries may span over multiple domains of interest.

The CrowdSearcher model is a simple form of data sharing between a conventional search engine and a social engine: the two environments communicate through selected search results which are produced by conventional engines and proposed as the input of the crowdsearching activity. The results of the previously mentioned questions could be shared with CrowdSearcher that supports the classical social actions of liking/disliking, tagging, expressin preferences, ranking. These tasks are not proposed to unknown crowds, but to trusted social network friends.

As users connections are in the order of hundreds, or even thousands on social networks, an accurate selection of people to whom ask is needed to improve the effectiveness of CrowdSearcher: if we were able to know the expertise level of each user about the question's topics, we could route each task to the right users. This thesis is focused on the *expert finding* problem for crowdsearching tasks: given a natural language query and a set of social network users, which is the best subset of users that have the knowledge required to perform the CrowdSearcher's tasks?. This main task leads to the following research questions:

- Can the analysis of social actions (e.g. posts, tweets, interaction with social groups, etc.) help in providing a better characterization of users for search tasks?
- Among the available approaches for expert finding, which one is better suited in the context of social networks?
- Are social networks oriented toward specific domains of expertise?
- Is the combined use of social network information useful to better characterize a user?
- Is the usage of semantic annotation techniques useful for improving the performance of an expert finding system?

ORIGINAL CONTRIBUTION

1. Piattaforma standard e condivisa per l'esecuzione dei task (Javascript)
2. Componenti modulari per una facilitarne la sotituzione/aggiurnamento
3. ???

OUTLINE

The thesis is organized in four main parts.

THE FIRST CHAPTER presents the fundamental aspects of crowd-based load distribution and the web enabling technologies used to implement such infrastructure and with wich benefits.

NEL SECONDO CAPITOLO viene desritto nel dettaglio il modello computazionale usato per la assegnazione/distribuzione dei task e gli attori che ne fanno parte.

NEL TERZO CAPITOLO vengono descritti gli use-case presi in considerazione fornendone i dettagli ed alcune di implementazioni possibili.

NELL'ULTIMO CAPITOLO viene descritta l'implementazione del modello descritto nei capitoli precedenti, sia sotto l'aspetto architetturale che di performance.

1 THE BACKGROUND

Recent years have seen an increasing interest in *Human Computation* and *Crowdsourcing* areas. One of the reason they are becoming so attractive is the growth of the Web. This has allowed to leverage the ability of people over the internet to perform tasks that even modern computers cannot achieve properly.

This chapter, first, focus on the key steps and developments in these fields that lead to the purposes of this thesis. We provide an overview of human computation and parasitic computing, then we introduce the technologies that enables the distributed computation on the web such as HTML5 for the task distribution and execution and WebCL for the task execution.

1.1 CROWD-BASED COMPUTATION DISTRIBUTION

Distributing computation (task computation) in the crowd means splitting the task execution into atomic subtask that can be executed by a host (human or not).

Write something about the crowd based distribution of the tasks, use references to (Mechanical turk Little *et al.*, 2010) if possible.

The online tool *MTurk* provide a framework for the creation distribution, execution and result gathering of task (called Human Intelligent Task (HIT)). Diring the creation a *Requester* The *Requester* can push request for executing HIT, these are

- Cos'è - FATTO?
- Crowdsource?
- come si fa (canali di distribuzione, assegnamento, controllo degli utenti)

1.1.1 Human computation

Computers are capable of performing many tasks, they can process large amounts of data and do billions of operation in a few seconds. However, there are still many problems that computers cannot solve or take too much time to solve even for the powerful pc.

Some of this are very simple tasks for humans, for example natual language processing and object regonition are hard to solve problem for a computer but natural for a human being, A great example for this kind of problem is recognizing hand-written text, even after years of research, humans are still faster and more accurate than ony computer.

Furthermore, there are problems that are too computationally expensive, such as many NP-complete problems like Traveling Salesman problem, scheduling problems, packing problems, and FPGA routing problems.

The expression *Human Computation* in the context of computer science is already used by Turing, 1950. However is Law and Ahn, 2011 to introduce the modern usage of the term. He defines human computation as a research area of computer science that aims to build systems allowing massive collaboration between humans and computers to solve problems that could be impossible for either to solve alone. But, in my opinion simple and direct definitions are better to get the point:

> *Some problems are hard, even for the most*
> *sophisticated AI algorithms.*
> *Let humans solve it...*
> — Edith Law

- ESP?
- reCAPTCHA?
- Crowdsourcing lo metto?

1.1.2 Parasitic computing

Parasitic computing[1] is a technique that, using some exploits and ad-hoc code, permits to execute computation on unaware host computer. This approach was first proposed by Barabási *et al.*, 2001 to solve the NP-complete 3-SAT problem using the existing TCP/IP protocol and its error handling routines.

Spiego meglio come funzionava il loro metodo?

Parassitic compiting has a strong relationship with *distributed computing*, in fact it is like a specialization of the general class of *distributed computing* where the user is unaware of the execution[2]. Given that we can list the main steps used to perform distributed computing:

- Split task into atomic operations executable by any host
- Send the code to all the host computers
- Execute the code
- Gather the results from the hosts
- Join all the hosts result and compute the task output

Distributed computing leverage on the idea of *divide and conquer* like the programming model of MapReduce[3]. Frameworks as Berkeley Open Infrastructure for Network Computing (BOINC) and Search for Extra-Terrestrial Intelligence *at* home (SETI@home) implement distributed computing paradigm to perform large scale operations (such as signal analisys) among the volunteers that installed the clients. These volunteers choose the project they are interested in and give the idle time of their machines to perform the computation.

Parasitic computing performs the same kind of task in the same *distributed* fashion but the main difference is that the users are unaware of the computation that is being executed on their pc.

[1] In this thesis we are not covering, neither we are interested, in the ethical or moral implication of using such programming model.

[2] In *distributed computing* the user can be unaware of the purpose computation is for or what actial code they are executing, but they are aware of the execution.

[3] Dean and Ghemawat, 2008.

- Differenza tra computazione parassitica e computazione distribuita (BOINC o seti@home) - FATTO?
- **Parlare di quante volte effettuiamo computazione parassitica senza sperlo.**
  Esempi?
- **Parasitic computing può anche essere fatto in un modo conscio.** Notificando all'utente la possibilità di eseguire del codice (senza sapere quale) in cambio di un ritorno di qualche tipo (Karame *et al.*, 2011).
- Using the same model of unaware host we can perform high level computation using JavaScript.*Modernizr*

The main drawback of distributed computing is the portability and distribution. The installation of some kind of client to execute the code can be seen as a problem for some user, as an example some users simply cannot install software on their workstation, due to security restriction or missing disk space. The other problem is distribution, the main purpose of these frameworks is to perform massive parallel computation, but for the computation to be really massive we need a lot of volunteers that installed the client on their pc and are online to execute the code.

Grafico con insiemi per distributed computing and parasitic computing?

PARASITIC JAVASCRIPT can lead to a solution of these problems using a widespread and standard technologies. Using the Web as the distribution platform the audiance can scale rapidly from to thousands to hundred thousands of users. Regarding the need of third part software installation and security issues, using JavaScript these problems are avoided, because all the code the browsers runs is executed into a sandboxed execution environment so it cannot harm the users pc. The same stands for the portability of the code, bacause almost all bowsers[4] support JavaScript with all the HTML5 features (see 1.2.1 on the following page), so the porting of the code is guaranteed on every system that can run a browser.

Let make an example **CREARE ESEMPIO CON BOINC E UN SITO DA 500.000 VISITE**

Using parasitc JavaScript can lead to some **hybrid** solution between distibuted and parassitic computing. Using the browser we can ask to user if it is willing to run some code [5] then we can proceed downloading all the required resource to run the code. This approach make possible to have a proactive approach to volunteer computing, so there is no more the need of waiting until the users are willing to spend some time running a task.

This **hybrid** approach is proposed in Karame *et al.*, 2011 as long as a µPayment model for task execution.

Spiego meglio il loro approccio?

- problema del distributed computing (installazione del client | distribuzione) - FATTO
- soluzione: piattaforma standard condivisa da tutti Javascript - FATTO
- problema HTML4 -> HTML5 collegamento - FATTO

[4] ***COUGH*** IE ***COUGH***

[5] **mettere una nota in cui si parla del revenue dell'utente e alla sezione in cui viene discusso meglio il tutto**

- permette una soluzione idriba (avviso che può essere eseguita della computazione, l'utente sceglie) - FATTO

1.2 ENABLING WEB-BASED DISTRIBUTED COMPUTATION

Web-based computation implies that a client is able to perform almost any kind of task that usually is done by an application software, as an example think about image analisys, audio/video playback or socket connection; these operations are available to developers without the need of additional libraries or external *plugins*.

When building Rich Internet Application (RIA) developers have to face the problem of building *rich* web application without the required tools for **communication**, **data access** and **data storage**. Access to raw data of images or audio, API for file management, data storage and full-duplex communication are all problems that could not be solved without using plugins like Flash or Silverlight.

The advent of HTML5 has brought a breath of fresh air to the Web. HTML5 specifies all these features as part of the language specifications so they are being implemented in all mayor javascript engines (Presto, V8, SquirrelFish, JägerMonkey). This means that almost all the required tools to build real *rich* internet application are built-in in the JavaScript language.

COMMUNICATION is being empowered by the introduction of *WebSocket* that enable full-duplex data exchange with the server. Also the introduction of Cross-origin Resource Sharing (CORS) give the developers the possibility to contact foreign servers using Asynchronous JavaScript and XML (AJAX) without the need of a proxy for forwarding the requests.

DATA ACCESS is obtained using HTML5 media elements (`<video>` and `<audio>`) or the File API.

DATA STORAGE is available through the `localStorage` and `sessionStorage` global variables or using IndexedDB or even a built-in WebSQL database.

With the introduction of all these features developers can use the power of JavaScript to perform image analysis, audio/video palyback (without any external plugin installed), create 2D/3D games and so on.

These features make possible to create tools like *Emscripten* that is a LLVM-to-JavaScript compiler. Basically allow developers to convert their C/C++ code into standard JavaScript, obviously the performance are not comparable but different level of code optimization lead to good performance gains in terms of code size and execution speed.

Additionally specification like CORS, not strictly related to JavaScript, allow the users to make cross-site request, that was a great limitation in JavaScript develpment.

1.2.1 HTML5

In this thesis when i refer to HTML5 i'm not speaking only about the HTML5 tag reference. I am speaking about a set of thechnologies and specifications related to HTML5. It includes the HyperText Markup Language version 5 (HTML5) specification itself, the Cascading Style Sheets (CSS3) recomendations and a whole new set of JavaScript APIs. So, first things first, lets make some clarification:

HTML5 refers to a new set of semantic tag (like `<footer>`, `<header>`, `<article>`, ...), media tags (like `<video>` or `<audio>`) and the so called Web Form 2.0.

CSS3 refers to the presentation layer specification including image effects, 3D transformation, tag selectors and form element validation.

JS refers to the new set of API provided, that enable interaction with all these new elements, and additional, non tag-related, functionalities (like WebSockets or WebWorkers).

With the advent of HTML5, like any new web-technology, many problems were resolved and many others have been created. The main issue with using HTML5 is the browser compatibility and browser-specific methods. Every borowser has its own implementation of the HTML5, this is mainly due to the early implementation of draft specification[6].

To avoid browser inconsistency we could use JavaScript frameworks. Frameworks like *jQuery* provide a layer of abstraction between browser-specific code and the user, giving developers JavaScript fallbacks for the most common API and additional features not covered by the standard implementation. Other tools like *Modernizr* give developers the ability to test if some HTML5 features are supported or not and provide a general fallback system for dynamically loading polyfills[7].

Now i will analyze in detail the main features of HTML5 to better understand their usefullness.

CANVAS Let's start with the official definition[8]

```
The canvas element provides scripts with a resolution-dependent
bitmap canvas, which can be used for rendering graphs, game graphics,
or other visual images on the fly.
```

So basically is a *Canvas*, like the name says, but give the developer the access to the raw pixel data of the canvas contents. Also in the canvas element you can draw the image taken from an `<img>` tag or a frame from a `<video>` tag. As you can se now we have the capability to manage image data directly and perform client-side task like image analisys or video manipulation. Obviously there are plenty of JavaScript libraries that give you methods to perform image filtering or generally image manipulation (like Pixastic or Camanjs), other libraries give you the possibility to create images on the fly (like Raphaël or Processingjs).

The canvas element also provide a 3D context to draw and animate[9] high definition graphics and models using the WebGL API. This API is mantained by the Khronos Group and is based on OpenGL ES 2.0 specifications.

[6] In fact HTML5 (at the time of writing) is not yet standardized, is still a draft. See `http://www.w3.org/TR/html5/`

[7] A polyfill is a JavaScript library or third part plugin that emulates one or more HTML5 features, providing websites to have the same *look and feel* also on older browser.

[8] Got from the specs: `http://www.w3.org/TR/html5/the-canvas-element.html#the-canvas-element`

[9] Animation is not natively supported, you must code it yourself.

On top of these API there are a lot of libraries[10] created for easy development, the most used is the Three JavaScript library, that ca be used for creating and animating 2D or 3D scenes in the canvas element.

WEBSOCKET The WebSocket is an API interface for enabling bi-directional full-duplex server communication on top of the Transmission Control Protocol (TCP) protocol. The WebSocket enables the clients to create a communication channel between the server and the client, allowing the server to push data to the clients and obtain *real* real-time content updates.

Like other HTML5 features, WebSocket has a library, build on top of the API, that provides easy access to these functionality as long as a couple of fallbacks. **socket** provide a single entry-point to create a connection to the server and manage the message exchange, it also provide a few fallbacks[11] to ensure cross-browser compatibility.

WEBWORKERS A problem you have to face when you are building computationally heavy JavaScript code is its single thread nature. Every script runs in the same thread, this can lead to some unwanted behaviour like browser freezing or the newly introduced warning dialog "*A script is slowing the browser*". The browser shows the dialog to prevent freezing of crashing of the whole bowser application, but this dialog prevent the script to fullfill their task. So how can we execute long running JavaScript computation if the browser stop the code?

Jenkin, 2008 proposed a timed-based programming structure that ensure the code to be run without any browser warning and also offer the developer to tweak the performance of the script by dynamiccaly adjusting the interval between the step execution. This method leaverage on the `setTimeout` function of javascript in order to split code into timestep-driven code chuncks to execute. Here is an example of loop translated into a time-based loop:

```
while condition do
    ...do something...
end
```

```
procedure STEP
    ...do something...
    if condition then
        setTimeout(STEP, delay)
    end
```

Obviously this is not a solution it is a way to hack the browser JavaScript performance monitor and avoid the warning dialog. WebWorkers provide a standard way to create *Workers* that execute in background, also performing heavy computation without harming the browser flow. Let's provide an official definition:

> The WebWorkers specification defines an API for running scripts in the background independently of any user interface scripts. This allows for long-running scripts that are not interrupted by scripts that respond to clicks or other user interactions, and allows long tasks to be executed without yielding to keep the page responsive.

So basically fills the gap of parallel code execution in JavaScript.

[10] For a reference see http://en.wikipedia.org/wiki/WebGL#Developer_libraries

[11] WebSocket, Adobe®Flash®Socket, AJAX long polling, AJAX multipart streaming, Forever Iframe,JSONP Polling

1.2.2 WebCL

With the advent of General-purpose computing on graphics processing units (GPGPU), the spreading of multicore CPUs and multiprocessor programming (like OpenMP) we can see emerging an intersection in parallel computing. This intersection is known as **heterogeneus computing**. Open Computing Language (OpenCL) is a framework for heterogeneus compute resources and so Web Computing Language (WebCL) is a porting of this technlogy to the web.

OpenCL uses a language based on C99[12] for writing *kernels*, functions that actually execute on OpenCL devices.

The main focus when building high-end web-application like 3D games is responsiveness. Altough JavaScript can be optimized and parallelized (see 1.2.1 on page 4) it cannot be fast as an application software, because JavaScript must be interpreted by the browser and then executed as machine code. WebCL provide an easy framework for building and running machine code in parallel directly from the browser.

- Come usiamo noi queste tecnologie
- task monitoring
- SIFT??

[12] A programming language dialect for the past C developed in 1999 (formal name ISO/IEC 9899:1999)

2 THE MODEL

When facing the problem of creating a suitable model for a task distribution system over the web we first need to think about the features our system must be able to perform. As we mentioned in the introduction we want to be able to perform task that are complex both in algorithmc and computational way, so we need a model able to manage both automatic and manual task computation.

In addition to this feature we want our model to be easily extendable with pluggable components defined during the task creation phase. The pluggability ensures that any extra computation can be added or can replace to the standard behaviour of the system.

The model we use can be separated in 3 cooperating submodels:

THE COMPUTATIONAL model describes the flow of the computation, from the task creation to the result gathering.

THE DISTRIBUTION model describes how a task can be distributed, to whom and what kind of steps are performed to check the result.

THE TASK AND PERFORMER model describes the lifecyvle of a task wrt the performer.

2.1 COMPUTATION MODEL

At first got task and then we improved it to be more general. We want more flexibility for our task, we need to create complex task that not only deliver code and gather results but can follow a non predefined workflow.

An example of a task one might want to perform is video time tagging. in this task mwe have a set of input data (the videos and we can also have a predefined set of availabe tags) and as output data we expect a set of tag/s for each time instant for each video.

To explain our model we split this task in these steps:

- Tag video (human+predefined)
- Verify video tag (human)
- Check good video (automatic)
- Repeat step 1 for the bad videos (automatic)

each step involve different data of the task (eg. step 1 operates on different selection from the main dataset, step 2 operates on a projection of the data). All these steps belong to the same **campaign** that is "*Video time tagging*", each step can be seen as a separate *task* with its input and output data, also each task must be distributed among users so it must be splitted into *subtasks* that insist on different[1] portions of the task data.

In out model we have a **Macro task**, that represent a instance of the *campaign*, and its a composition of **Task**, that may have dependencies between them, at least we have **Micro task** that represents the *subtasks*.
many ways like integrate in the task flow a verification step

2.2 TASK DISTRIBUTION MODEL

TaskDistributionModel

2.3 TASK AND PERFORMER MODEL

Task+Performer Model

2.4 CROWDSEARCHER????

CrowdSearcher

[1] The data they insist on are selection from the task input data, with or without overlapping

3 THE USE-CASES

Speak about the use cases chosen and the category they belong to.

3.1 HUMAN-DRIVEN

Dato un testo disambiguarlo usando YAGO (AIDA, https://d5gate.ag5.mpi-sb.mpg.de/webaida/), EntityPedia?, e altri *Modernizr*

3.2 MACHINE-DRIVEN

Machine driven Scale-Invariant Feature Transform (SIFT)

3.3 HUMAN AND MACHINE DRIVEN (HYBRID)

Hybrid (Face recognition)

4 IMPLEMENTATION AND EVALUATION

4.1 ARCHITECTURE

4.2 PERFORMANCE COMPARISON???

A CONCLUSION AND FUTURE WORKS

ACRONYMS

HTML5 HyperText Markup Language version 5

HTML5 is a markup language for structuring and presenting content for the World Wide Web, and is a core technology of the Internet originally proposed by Opera Software.

WebCL Web Computing Language

The WebCL working group is working to define a JavaScript binding to the Khronos OpenCL standard for heterogeneous parallel computing. WebCL will enable web applications to harness GPU and multi-core CPU parallel processing from within a Web browser, enabling significant acceleration of applications such as image and video processing and advanced physics for Web Graphics Library (WebGL) games.

SIFT Scale-Invariant Feature Transform

SIFT is an algorithm in computer vision to detect and describe local features in images.

OpenCL Open Computing Language

OpenCL is a framework for writing programs that execute across heterogeneous platforms consisting of CPU, GPU, and other processors. OpenCL includes a language (based on C99) for writing *kernels* (functions that execute on OpenCL devices), plus APIs that are used to define and then control the platforms. OpenCL provides parallel computing using task-based and data-based parallelism.

WebGL Web Graphics Library

WebGL is a cross-platform, royalty-free API used to create 3D graphics in a Web browser. Based on OpenGL ES 2.0, WebGL uses the OpenGL shading language, GLSL, and offers the familiarity of the standard OpenGL API. Because it runs in the HTML5 Canvas element, WebGL has full integration with all DOM interfaces.

CORS Cross-origin Resource Sharing

Cross-origin resource sharing (CORS) is a web browser technology specification which defines ways for a web server to allow its resources to be accessed by a web page from a different domain. Such access would otherwise be forbidden by the same origin policy. CORS defines a way in which the browser and the server can interact to determine whether or not to allow the cross-origin request. It is a compromise that allows greater flexibility, but is more secure than simply allowing all such requests.

RIA Rich Internet Application

Rich Internet Applications (RIA) are web-base application taht have many of the characteristics of desktop application software.

HIT Human Intelligent Task

TCP	Transmission Control Protocol
AJAX	Asynchronous JavaScript and XML
CSS3	Cascading Style Sheets
BOINC	Berkeley Open Infrastructure for Network Computing
GPGPU	General-purpose computing on graphics processing units
SETI@home	Search for Extra-Terrestrial Intelligence *at* home SETI@home is an Internet-based public volunteer computing project employing the BOINC software platform, hosted by the Space Sciences Laboratory, at the University of California, Berkeley, in the United States. Its purpose is to analyze radio signals, searching for signs of extra terrestrial intelligence, and is one of many activities undertaken as part of SETI.

Essential bibliography

Barabási, A.L., V.W. Freeh, H. Jeong, and J.B. Brockman

2001 "Parasitic computing", *Nature*, 412, 6850. (Cited on p. 2.)

Bozzon, A., M. Brambilla, and S. Ceri

2012 "Answering search queries with CrowdSearcher", in *Proceedings of the 21st international conference on World Wide Web*, ACM. (Cited on p. xi.)

Dean, J. and S. Ghemawat

2008 "MapReduce: Simplified data processing on large clusters", *Communications of the ACM*, 51, 1. (Cited on p. 2.)

Group, Khronos OpenCL Working *et al.*

2008 "The opencl specification", *A. Munshi, Ed.*

Jenkin, N.

2008 "Parasitic JavaScript". (Cited on p. 6.)

Karame, G.O., A. Francillon, and S. Čapkun

2011 "Pay as you browse: microcomputations as micropayments in web-based services", in *Proceedings of the 20th international conference on World wide web*, ACM. (Cited on p. 3.)

Law, E. and L. Ahn

2011 "Human computation", *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5, 3. (Cited on p. 1.)

Little, G., L.B. Chilton, M. Goldman, and R.C. Miller

2010 "Turkit: human computation algorithms on mechanical turk", in *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM. (Cited on p. 1.)

Quinn, A.J. and B.B. Bederson

2011 "Human computation: a survey and taxonomy of a growing field", in *Proceedings of the 2011 annual conference on Human factors in computing systems*, ACM, pp. 1403–1412.

Turing, A. M.

1950 *Computing Machinery and Intelligence*. (Cited on p. 1.)

WebGL-OpenGL, ES

2011 "2.0 for the Web", *Verkkodokumentti< http://www. khronos. org/webgl/>. Luettu*, 16.

Online resources

Emscripten 2012 , http://emscripten.org/. (Cited on p. 4.)

jQuery 2012 , http://www.jquery.com/. (Cited on p. 5.)

Modernizr 2012 , http://modernizr.com/. (Cited on pp. 3, 5, 11.)

MTurk 2012 , http://www.mturk.com/. (Cited on p. 1.)

Socket.io 2012 , http://socket.io/.