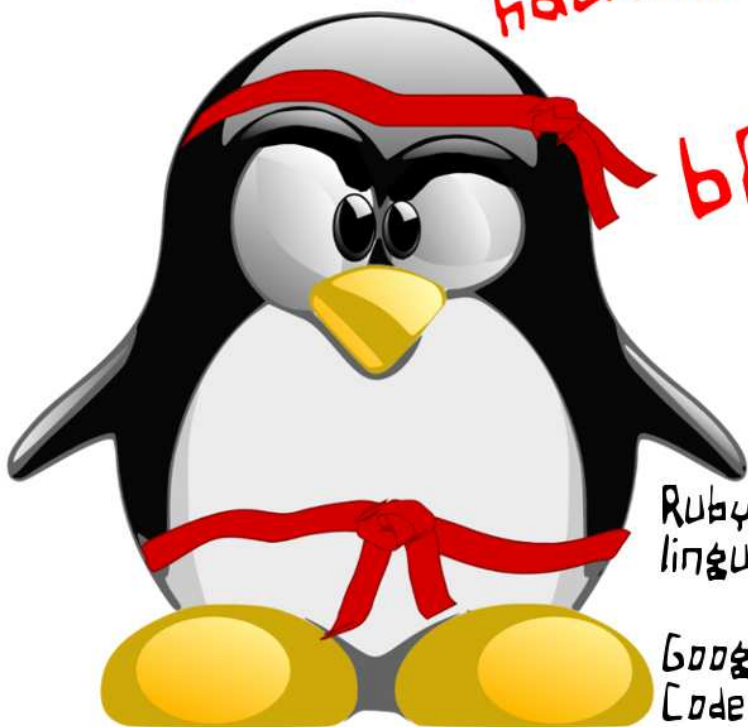


Giorna Linux 2.0

just keep on hacking!



БАНЗАИ!

Ruby: una gemma di
linguaggio

Google Summer of
Code

Open Source
Computer Vision

Maneggiare con CUDA



Opportunità al Politecnico

Gennaro Florino <rinolfo@gmail.com>

IN QUESTO ULTIMO ANNO, cioè durante il periodo in cui ho ricoperto la carica di segretario del POuL, ho avuto modo di osservare ed interagire con molteplici aspetti del Politecnico di Milano e ho potuto notare come si stia evolvendo la struttura burocratica dell'ateneo in una forma più leggera e flessibile dal punto di vista amministrativo.

“Gestionalmente” parlando, questo ha portato molti vantaggi sia al Politecnico come struttura, che nei tagli di personale imposti ha dovuto trovare una rapida soluzione, sia soprattutto agli studenti che si interfacciano con i suoi servizi.

In particolare il Politecnico oltre alla didattica offre altre interessanti possibilità agli studenti, i quali spesso e volentieri non immaginano neanche di poter usufruire di fondi stanziati dal Politecnico stesso o magari non osano incominciare l'iter burocratico per timore delle eventuali difficoltà. I Servizi Generali agli Studenti si occupano proprio di questo: tramite l'apposito sito *eventistudenti* viene indetto ogni anno a novembre un bando in cui vengono vagliate tutte le proposte degli studenti ed allocati i fondi a disposizione.

Il mio consiglio è, se si ha una idea, di pro-

vare a chiedere: si può scoprire che dall'università si possono ottenere anche più dei 300 CFU della laurea e noi come POuL ne siamo un esempio. Nel POuL infatti non realizziamo solo eventi, ma anche esperienza, amicizia e cooperazione, tutte cose che la didattica ordinaria non concede. Per questo torno al mio invito ad informarvi sulle opportunità del Politecnico e a non spaventarvi, poiché le procedure si sono notevolmente snellite.




Se avete domande particolari, naturalmente metto a disposizione l'esperienza che ho maturato come nel migliore degli spiriti Open Source.



Indice

Ruby: una gemma di linguaggio	3
Google Summer of Code	6
Open Source Computer Vision	8
Maneggiare con CUDA	13

Quest'opera è rilasciata sotto la licenza Creative Commons BY-NC-SA 2.5. Questo significa che sei libero di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire o recitare l'opera e creare opere derivate alle seguenti condizioni:

-  **Attribuzione.** Devi riconoscere il contributo dell'autore originario.
-  **Non commerciale.** Non puoi usare quest'opera per scopi commerciali.
-  **Condividi allo stesso modo.** Se alteri, trasformi o sviluppi quest'opera, puoi distribuire l'opera risultante solo per mezzo di una licenza identica a questa.

In occasione di ogni atto di riutilizzazione o distribuzione, devi chiarire agli altri i termini della licenza di quest'opera. Se ottieni il permesso dal titolare del diritto d'autore, è possibile rinunciare ad ognuna di queste condizioni.

Le tue utilizzazioni libere e gli altri diritti non sono in nessun modo limitati da quanto sopra.

Questo è un riassunto in linguaggio accessibile a tutti del Codice Legale:

<http://creativecommons.org/licenses/by-nc-sa/2.5/it/legalcode>



Ruby: una gemma di linguaggio

Simone D'Amico <sim@me.com>

Dall'Oriente con furore

ようこそ!

(Benvenuti!) In questa serie di articoli vi farò conoscere le potenzialità di Ruby, il miglior amico di un programmatore.

Ruby nasce nel 1993 nel Sol levante dalla fervida mente di Yukihiro "matz" Matsumoto. Ruby fonde le caratteristiche dei linguaggi preferiti da matz (Lisp, Ada, Perl, Smalltalk, Eiffel, Ada) in un unico linguaggio, bilanciando programmazione funzionale e imperativa. Nel 1995 matz rilascia gratuitamente l'interprete sotto licenza open source e conquista subito uno zoccolo duro di programmatori in terra nipponica.

L'anno della svolta è però il 2006: con la diffusione di potenti framework, Ruby raggiunge un bacino di utenza mondiale. Vale la pena ricordare Nitro e Rails per lo sviluppo di applicazioni Web; in particolare, il secondo è diventato lo standard di riferimento per framework MVC di altri linguaggi.

Un fulgido esempio di applicazione Rails è l'applicazione di social networking Twit-

ter, giunta all'onore delle cronache per le vicende iraniane. Un altro framework molto noto è Metasploit, usato anch'esso per altri nobilissimi scopi.



Il kit del piccolo minatore

L'interprete Ruby è disponibile per moltissimi sistemi operativi, e i programmi, essendo non compilati, girano senza troppi problemi su qualsiasi piattaforma supportata.

Il kit del minatore si compone essenzialmente dei seguenti programmi:

- *ruby*: l'interprete dei nostri comandi;
- *irb*: una shell interattiva estremamente utile (specialmente durante l'apprendimento del linguaggio);

- *ri*: il programma per visualizzare la documentazione (ci sono oltre 9000 metodi nella libreria standard) estrapolandola direttamente dai sorgenti dei programmi;
- *gem*: l'apt delle librerie Ruby (in gergo chiamate "gemme"), ha il compito di installarle e mantenerle aggiornate.

Ecco brevemente come potete ottenere Ruby:

- utenti Windows: cercate su Google "Ruby one click installer" e troverete un installer wizard con dentro tutto, se avete problemi arrangiatevi :-)
- utenti Linux (distribuzioni Debian-based): date da shell il comando
sudo apt-get install Ruby1.9 libRuby1.9 libreadline-Ruby1.9 irb1.9 Rubygems
- utenti Mac OS X: Ruby in casa Apple deve essere molto apprezzato in quanto incluso di default da Tiger in poi; suggerisco tuttavia di installare l'ultima versione attraverso MacPorts (www.macports.org) con il comando da terminale
sudo port install Ruby19

Per le altre piattaforme (e gli utenti esperti) consiglio la compilazione dei sorgenti da SVN, potrete così godere in anticipo delle novità introdotte dagli sviluppatori. Altro strumento essenziale per un minatore provetto è un buon editor. Per l'editing di sorgenti Ruby potete usare il vostro editor preferito, tuttavia gli utenti Mac hanno una marcia in più grazie a Textmate, l'editor Ruby (e non solo) per eccellenza, purtroppo a pagamento e

closed source, ma ampiamente espandibile grazie ai plugin della comunità. Per le altre piattaforme dovrebbero esserci in circolazione cloni di buona fattura.



Perché il Rubino luccica?

Ecco perché preferire Ruby nella bolgia dei linguaggi di scripting.

1. In Ruby TUTTO è un oggetto. Persino la costanti numeriche sono oggetti. Sì, anche la vostra ragazza è un oggetto, ma non usatela come tale senza una corretta gestione delle eccezioni.
2. Ruby è estremamente flessibile. La sintassi è molto vicina al linguaggio parlato. Se vi stato chiedendo se il programma potrebbe funzionare scritto in quel modo, probabilmente lo farà (Ruby è noto per essere un linguaggio anti-Murphyano).
3. Il codice prodotto è estremamente chiaro e le convenzioni di nomenclatura di classi e variabili rendono molto facile la loro identificazione. In realtà classi e variabili sono concetti fittizi, in quanto estendibili "alla c.d.c." durante tutto il programma (anche a run-time).
4. Le closures (i blocchi) sono la parte funzionale di Ruby presa direttamente da Lisp. Nei prossimi articoli avremo modo di ammirarne l'estrema potenza.

5. Il ducktyping: “sembra una papera e starnazza, deve essere una papera”. Ovvero Ruby non è fortemente tipizzato, ma un oggetto è distinto in base ai messaggi che può ricevere e non al suo tipo.
6. I “mixin”, ovvero l’eredità multipla resa semplice e utile. Basta includere un pacchetto di metodi in diverse classi affinché rispondano agli stessi messaggi.
7. Gestione delle eccezioni, come si conviene a un buon linguaggio OOP.
8. Garbage collector, per lo stesso motivo.
9. Facilità di estensione attraverso il C: sotto il cofano avrete la potenza delle migliaia di librerie scritte dalla community.
10. Chicca finale: il multi-threading indipendente dalla piattaforma, disattivato nel caso la piattaforma di esecuzione non lo supportasse (ad esempio sotto DOS).

quello che noi Rubysti chiamiamo il piccone (*pickaxe*), un ottimo libro della Pragmatic Bookshelf.

Per ora è tutto. Alla prossima puntata e... attenti agli infidi pitoni nelle miniere!

Fonti

- <http://it.wikipedia.org/wiki/Ruby>
- <http://www.ruby-lang.org>
- *Programming Ruby 1.9: The Pragmatic Programmers' Guide* (aka *PickAxe*), Dave Thomas, Chad Fowler, Andy Hunt

Ok, diamoci da fare!

Aspetta... non mi pagano abbastanza per scrivere così tanto in così poco tempo :-)

Intanto che aspetti ansiosamente il prossimo numero del Giornalinux, puoi sempre dare un’occhiata alle infinite risorse su Internet. Consiglio in particolare il tutorial interattivo online completabile in 15 minuti (<http://tryRuby.sophrinix.com>), con cui potrai darti le basi del linguaggio. Nel caso fossi un feticista della carta (come il sottoscritto), un acquisto imprescindibile è

Google Summer of Code

Alessandro Sivieri

<alessandro.sivieri@ieee.org>

UNA PREMESSA è necessaria: questo articolo non vuole essere una pubblicità di Google; l'iniziativa che vi presenterò è, secondo me, un ottimo modo per coinvolgere gli studenti nello sviluppo dei migliori progetti Open Source esistenti.

L'articolo si rivolge soprattutto a coloro che conoscono linguaggi di programmazione ed a cui piacerebbe sviluppare codice libero: non sono necessarie conoscenze specifiche nè grandi capacità di programmazione o requisiti di lavoro in team, solo una buona inventiva su come realizzare quanto proposto in uno dei singoli progetti.



Boost C++ Libraries, Debian, Eclipse Foundation, Fedora Project, GCC, Gnome, GIMP, OpenOffice, KDE, Mozilla, MySQL, Python Software Foundation, Tor,

The Linux Foundation, Wordpress: sono solo alcune delle organizzazioni che sviluppano software libero, e sono solo alcune di quelle che partecipano al bando del *Summer of Code*.

Panoramica

Google organizza tutti gli anni, a partire dal 2005, la Summer of Code: molte delle organizzazioni più famose che sviluppano software libero, ed anche alcune meno famose, propongono agli studenti alcune idee di possibili estensioni dei propri applicativi, questo in genere a marzo; chi desidera partecipare al bando, necessariamente uno studente di qualche grado, può scrivere una proposta di implementazione di una o più idee ed inviarla a Google.

I team che, all'interno delle singole organizzazioni, si occupano della Summer of Code, vagliano le proposte e decidono quali sono le migliori come possibilità di realizzazione nei tre mesi estivi, come inventiva, come capacità dello studente che ha fatto la proposta (ad esempio, aver già collaborato con quel progetto o conoscere alcune delle tecnologie impiegate potrebbe aiutarvi). Al termine, Google pubblica l'elenco degli studenti che hanno superato il con-

corso; quest'anno ne sono stati scelti circa 1000, provenienti da tutto il mondo.

Lo sviluppo

A questo punto, avete un mese circa per conoscere la community dell'organizzazione con cui partecipate e per iniziare a progettare la vostra proposta e/o a conoscere le tecnologie coinvolte.

Nei tre mesi da metà maggio circa a metà agosto inizierà lo sviluppo vero e proprio, e stipendiato direttamente da Google: sì, avete capito bene, venite pagati per contribuire a codice Open Source (ed ai miei detrattori dirò: ma non dicevate che non si guadagnava con il codice libero?).

Ci sono due valutazioni che il mentore, cioè la persona dell'organizzazione che vi segue e vi aiuta nello sviluppo, deve compilare sul vostro lavoro, e se superate entrambe avrete superato la Summer of Code, contribuito a codice usabile da tutto il mondo, ed avrete messo da parte qualche soldo.

Un appunto importante: il codice prodotto non apparterrà a Google, ma a voi, e verrà pubblicato secondo le licenze normalmente in uso dall'organizzazione per cui collaborerete.

La mia esperienza

Io ho partecipato, per la prima volta, al Summer of Code quest'anno, per un progetto di KDE (www.kde.org), ed è stata un'esperienza decisamente molto interessante: collaborare con persone da tutto il mondo (il mio mentore era tedesco, mentre un altro ragazzo che lavo-

rava per un progetto simile era statunitense), la possibilità di conoscere le altre persone che lavorano all'interno di KDE, che ho incontrato a novembre durante un meeting dell'organizzazione stessa, con cui sto continuando a collaborare.



La possibilità di vedere il proprio codice distribuito assieme a quello di tanti altri sviluppatori, la soddisfazione di aver contribuito per quel software che avete usato per anni; vedere nella mailing list ufficiale degli studenti GSoC discussioni informatiche tra americani, francesi, indiani, iraniani, cinesi, italiani, sudafricani e più in generale da tutto il mondo, senza pregiudizi o scontri di qualunque tipo, non ha prezzo.

In conclusione

Partecipate, partecipate, partecipate: è un'occasione unica per poter entrare nel mondo del software libero non più da meri utilizzatori, ma ora anche da programmatori; sono certo che tra le decine di idee proposte dalle decine di organizzazioni, ce ne sarà almeno una per il vostro linguaggio preferito, per il vostro progetto preferito, per la vostra tecnologia preferita.

Per altre informazioni potete vedere qui:

- <http://socghop.appspot.com>
- <http://code.google.com/soc>
- http://en.wikipedia.org/wiki/Google_Summer_of_Code

Perciò: preparate la vostra application al concorso, ed in bocca al lupo!

Open Source Computer Vision

Laura Camellini <jilt@inventati.org>
Francesco Savignago
<ctorctor@gmail.com>

OPEN SOURCE COMPUTER VISION (OpenCV) è il nome di un progetto pubblicato dalla Intel come libreria di funzioni di programmazione per la computer vision in real time (<http://opencv.willowgarage.com/wiki>). Tra le applicazioni più esemplificative di queste funzioni troviamo l'identificazione di oggetti, la segmentazione e il riconoscimento di volti e di gesti, il tracciamento del movimento della telecamera e degli oggetti, la robotica di movimento e molte altre. La parte più interessante di OpenCV risiede per noi nel riconoscimento e tracciamento automatico dei volti.

Storia e progetti in corso

OpenCV a livello di codice paga il prezzo di uscire da un'azienda come la Intel, che ne ha influenzato non poco la struttura e ne influenza la vita attuale a livello di progetto. La Google summer of Code ha rifiutato infatti i finanziamenti che tutti consideravano scontati e spesso gli sviluppatori, abituati a gestirsi in maniere

più "comunitarie", trovano poco pratico approcciarsi ad un codice con le modalità imposte dalla Intel; per questo motivo sono nati progetti paralleli a OpenCV, come IVT (<http://ivt.sourceforge.net>). Sorvolando sull'annosa questione dei finanziamenti (sempre attuale, chiarissimi) sono molti i progetti "cornice" che sfruttano il potenziale di questa libreria, alcuni rivolti ad uno sfruttamento embedded delle sue funzioni (quindi in locale), altri rivolti al processing di immagini da remoto attraverso la rete; si lavora sia sull'hardware come per Gumstick (www.gumstick.com), un progetto commerciale di single board computer specifici per l'uso di certe funzioni tra cui quelle di OpenCV (<http://danielbaggio.blogspot.com/2009/01/compiling-opencv-for-gumstix.html>), sia sull'integrazione alle interfacce software più utilizzate come Blender (www.blender.org), in cui si riesce a tracciare la posizione della testa grazie ad OpenCV per restituire il giusto punto di vista in una scena 3D, per esempio in un videogioco.

Esistono poi dei progetti paralleli, come CCV (*Community Core Vision*, <http://ccv.nuigroup.com>) che si propone di competere con OpenCV nei

campi del tracciamento di oggetti e del “machine sensing”; gli sviluppatori di CCV hanno partecipato alla Google Summer of Code con un progetto di tracciamento delle mani e la loro idea è stata finanziata nonostante la presenza di un progetto più grosso come OpenCV: questo dovrebbe per lo meno far pensare riguardo alle scelte di finanziamento del colosso dei metadati. Videoman(ager) (<http://videomanlib.sourceforge.net>) si propone invece di gestire uno o più flussi video da combinare con oggetti grafici anche tridimensionali; con Videoman è possibile usare la libreria OpenCV nativa sui flussi video e anche tracciare in 3D gli oggetti nei video.

Ultimo progetto parallelo ad OpenCV, ma non meno importante degli altri, è Torch3Vision (<http://torch3vision.idiap.ch>), una libreria con licenza BSD basata su Torch, con funzioni che fanno parte anche del progetto OpenCV; ultimamente gli algoritmi di face detection sono stati aggiornati e semplificati per rendere possibile il tracciamento di più volti in una stessa immagine. Torch3Vision è un progetto che dimostra la volontà di molti ricercatori di poter usare strumenti Open Source per sviluppare ricerche e rendere disponibili e utilizzabili al grande pubblico del mondo digitale le funzioni base di visione e riconoscimento, al di là delle intenzioni dei grandi marchi del mondo digitale come Intel o Google.

GStreamer

Anche GStreamer (<http://gstreamer.org>) attualmente usa OpenCV con fi-

nalità e modalità interessanti, come si può vedere nel repository creato da Michael Sheldon (<http://github.com/Elleo/gst-opencv>), anche lui studioso di processing di immagini in robotica.

GStreamer è un insieme di librerie che comprende componenti per la gestione della grafica e dei media audiovideo; viene usato in particolare per il processing e lo streaming di video e audio, supporta sia il mixaggio dell'audio sia l'editing non lineare del video. Il progetto è stato concepito in modo da fornire un core funzionale su cui ogni programmatore può caricare codec e filtri come plugin. GStreamer è utilizzabile su tutti i sistemi operativi, processori e compilatori e gestisce i plugin dinamicamente, con un container di funzionalità caricate a partire da un oggetto condiviso come modulo; il nome del contenitore è *GstPlugin*. Lo strumento da linea di comando *GstLaunch* permette di creare e gestire velocemente i prototipi di pipeline audio-video ma non ancora di salvarli in formato XML.

L'API di GStreamer è accessibile attraverso C e Python, ma la parte più interessante di questo progetto sono proprio i plugin, divisi in categorie qualitative, dove c'è spazio anche per le libere sperimentazioni. I plugin sono divisi in “classi” o categorie: le “prime classi” vengono seguite e aggiornate e sono considerate funzionalità che non possono mancare al software, le classi “ugly” e “bad” sono invece composte da pezzi di codice quasi grezzi, ancora da provare o da completare. Il set di plugin pubblicato da Michael Sheldon si propone di unire la praticità di GStreamer nel codificare video di formati diversi con il potenziale di

OpenCV, andando a cercare quali funzionalità potrebbe regalare al nostro PC l'uso di algoritmi che processano immagini nel mondo della robotica.

OpenCV e il Web

Cosa sarebbe il nostro PC senza il Web? GStreamer e OpenCV sono concepiti per fornire un core di funzionalità da sviluppare a seconda dei fini specifici a cui è dedicata l'interfaccia sovrapposta. Sono già tanti i progetti che usano le funzionalità di GStreamer per i motivi più svariati, come Pitivi (<http://pitivi.org>) per l'editing video, e Amarok (<http://amarok.kde.org>) per l'audio, per citare solo i più popolari.

Noi però siamo nell'epoca del Web 2.0. Adesso, per noi, l'interfaccia può avere un valore molto diverso: può diventare strumento di social networking, ad esempio per raccogliere ed elaborare dati audiovisivi dagli utenti di una rete. Ormai non pensiamo più all'interfaccia del software di editing video usata dall'artista per fare il suo cortometraggio; oggi anche il figlio di 8 anni della casalinga di Voghera vuole mettere il video del suo gol di domenica scorsa al campetto su Youtube. Ma se invece di collezionare solo filmati dei gol Youtube si preoccupasse di proteggere la privacy di questi utenti, nel caso non vogliano vedere le proprie facce pubblicate? Se tutti potessero pubblicare i propri contenuti coprendo i volti delle persone riprese grazie ad un metodo semplice come una pipeline di GstLaunch e OpenCV (che, essendo librerie libere, sono accessibili ad ogni utente), avremmo un piccolo strumento di emanci-

pazione digitale che aiuta l'utente a capire e manipolare l'interfaccia che utilizza nella maniera che preferisce, e lo educa al rispetto degli altri.

Un tentativo del genere pare lontano, purtroppo, dai toni con cui vengono pubblicizzate le interfacce di social networking più usate al momento (cito a memoria: "Sparla anche tu dei tuoi amici su Facebook"), e certamente rimane lontano dagli intenti con cui dette interfacce sono state costruite e fatte funzionare, ma quello di salvaguardare la propria privacy rimane un bisogno spesso insoddisfatto degli utenti stessi. Spesso nelle situazioni che ne derivano si generano dispute lunghissime e articolate: cosa succederà se permetteremo l'uso degli strumenti che abbiamo a disposizione come mezzo per controllare e non per proteggere l'anonimato dei cittadini?

Il rischio rimane sempre quello del vecchio demone del cyberpunk, la macchina che supera l'uomo per sopraffarlo. L'idea di emancipazione digitale accennata prima vuole essere un tentativo di democratizzazione degli strumenti avanzati che le tecnologie mettono a nostra disposizione, in questo caso per dare la possibilità ad ognuno di rendere anonimi i video pubblicati online, in un mondo in cui le telecamere di ultima generazione e gli spider che fiutano le pagine Web hanno già ampiamente cominciato a fare uso di queste stesse tecnologie come un esercito di piccoli robot che marcia seguendo la scia delle nostre immagini.

Un obiettivo del genere appare lontano anche dalle intenzioni delle nostre istituzioni; basti pensare a progetti come Indect ("Intelligent Information System Suppor-

ting Observation, Searching and Detection for Security of Citizens in Urban Environment” <http://www.indect-project.eu>, un progetto che punta ad usare le stesse tecnologie per finalità totalmente opposte. Gli obiettivi principali vengono elencati nel sito:

- sviluppare una piattaforma per la registrazione e lo scabio di dati operativi, l’acquisizione di contenuti multimediali, il processing “intelligente” e la ricerca sistematica e il riconoscimento di comportamenti strani o violenti;
- sviluppare il prototipo di un sistema integrato sulla rete, per dare supporto alle operazioni della polizia, fornendo tecniche e strumenti per l’osservazione di vari oggetti mobili;
- sviluppare un motore di ricerca che combini la ricerca diretta d’immagini e video basata sui watermark e un archivio di metadati sotto forma di watermark digitali.

La discussione etica viene opportunamente affrontata nel sito, ma fondamentalmente lasciata aperta. Ci sarebbe da riflettere alla luce delle differenti leggi che in ogni stato regolano la materia, riguardo al ruolo che questa figura del cittadino sta assumendo all’interno della nostra struttura sociale, in continua evoluzione grazie alle nuove tecnologie a disposizione e grazie agli eventi storici di massa più importanti di cui siamo protagonisti, come l’immigrazione dalle zone più povere del mondo. Il cittadino europeo del 2000 avrà la possibilità di usufruire dell’avanzamento tecnologico della propria era in maniera davvero consapevole, o sarà

vittima di una tecnologia in mano a pochi che aggrega il potere e le informazioni solo nelle loro mani? Le tecnologie adesso sono disponibili online grazie alla diffusione di Internet, quindi il gioco di forza ormai rimane solo nella maniera in cui saranno usate e si diffonderanno.

A che punto siamo?

Uno strumento per coprire le facce con OpenCV è già presente nel repository di Michael Sheldon; esso però si ferma alla funzionalità di tentare, sulla base di alcuni parametri chiamati “cascade”, di riconoscere i volti su uno spazio di colore e di coprirli con un semplice blur.

La pipeline generica per usare il plugin con GStreamer è la seguente:

```
gst-launch-0.10 filesrc location=test.avi
! decodebin name=decode decode. !
queue ! ffmpegcolorspace ! faceblur !
ffmpegcolorspace ! theoraenc ! oggmux
name=mux decode. ! queue ! audiocon-
vert ! vorbisenc ! mux. mux. ! filesink
location=test-blurred.ogv
```

Questo filtro specifico, alla cui creazione il progetto è finalizzato (http://jilt.indivia.net/docu/doKu/doku.php?id=projects:web_tool), non è ancora messo a punto perfettamente in quanto non riconosce tutti i volti dei video e spesso non segue quelli riconosciuti; ne risulta che l’anonimizzazione fa “cilecca”, per cui, per ottenere un risultato ottimale in più tipi di immagini, sono ancora necessarie delle sperimentazioni per lo studio dei parametri di ogni funzione usate in ogni specifica evenienza. Nella wiki dedicata al progetto si trova

una piccola trattazione teorica: il progetto è chiamato Catseye (“Occhio di Gatto”), come la pietra preziosa molto usata in gioielleria, ma anche per alludere ad una sfera privata come quella di cui fanno parte gli animali domestici.

Riuscire a far utilizzare un filtro del genere dall’“utente medio” potrebbe essere un grosso passo avanti nell’umanizzazione e riadattamento degli strumenti Web 2.0 alla società civile: questi esperimenti sono quelli che, grazie alle possibilità fornite oggi dalle licenze aperte, si preoccupano di far girare lo strumento digitale intorno all’uomo ed alle sue esigenze, e non l’uomo intorno alle esigenze di accumulo di dati e profili delle dinamiche digitali attuali.

Esiste un altro progetto che riunifica la funzionalità di riconoscimento del “Cascade” di OpenCV con il “Camshift” della stessa libreria, dando modo alla macchina di riconoscere il volto per poi tracciarne il movimento seguendolo nel video; anche questo è un progetto piccolo e di scarsa funzionalità per adesso, che può essere rintracciato insieme al suo autore Robin Hewitt nel giornale online di robotica “Cognotics” (http://www.cognotics.com/opencv/downloads/camshift_wrapper).

Il wrapper è stato scritto per Microsoft Windows, ma sarebbe utile tentare una nuova stesura del codice per Linux per implementare nuovamente Catseye di cui si parlava nei paragrafi precedenti. Le modalità usate dal wrapper per combinare riconoscimento e tracciamento sono anch’esse da ridiscutere, ma il piccolo esempio può dimostrare a tutti che il mondo dell’Open Video si sta definendo

in una maniera tale da lasciare aperte le direzioni di sperimentazione, date dalle funzionalità di base ormai avanzate, sulle interfacce e sul loro impatto sociale. Sarà sulle convinzioni umane quindi, sulle condizioni di utilizzo del computer e sulle esigenze umane su cui è costruito il momento di vita digitale che si giocherà l’impatto di queste tecnologie. Finalmente torniamo all’uomo: dai tecnicismi a cui abbiamo dedicato ampio spazio nel nostro articolo, si svolgerà qui, alla fine, la parte più interessante di questa storia.

Maneggiare con CUDA

Saten <satene.r@gmail.com>

AL GIORNO D'OGGI siamo abituati ad usare il computer così spesso che probabilmente molti di noi non ci fanno caso (troppo alla lontana...).

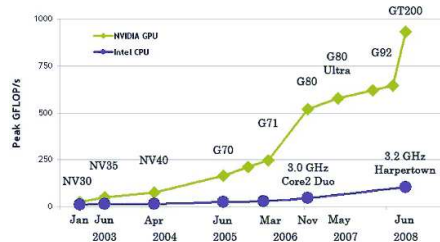
Chi non ha mai sentito parlare di CPU? La CPU (*Central Processing Unit*) è il componente più noto di un computer, in quanto è quello che svolge i calcoli e le istruzioni. Quando qualcuno desidera un PC più potente, spesso sospira su quanto sarebbe bello poter cambiare il processore... e perché no, fare un bell'upgrade della RAM!

Una sigla meno nota è invece GPU, che sta per *Graphical Processing Unit*. Possiamo considerare la GPU come il processore che si trova sulla scheda video. Le operazioni che si svolgono con la scheda video, infatti, sono in genere così complesse da richiedere un processore dedicato.



I progressi in questo campo sono stati

così grandi che la potenza di una GPU moderna, in termini di pura velocità, supera anche di tantissimo quella di una CPU.



Qualcuno potrebbe chiedersi perché, al di là di intuibili ragioni di costi, non si usino le GPU al posto delle CPU se sono così performanti. La ragione è che le GPU richiedono situazioni particolari per essere sfruttate al meglio, e in particolare necessitano di operare in parallelo su istruzioni omogenee; in questo modo si sfruttano i molti processori che costituiscono una GPU ma che non possono operare indipendentemente uno dall'altro per motivi architetturali.

CUDA

La nVidia, nota casa produttrice di schede video, ha avuto per prima l'intuizione

di distribuire dei driver, un SDK (*Software Development Kit*) ed un toolkit per sviluppare programmi pensati per sfruttare la potenza di calcolo delle sue GPU: è nato così il progetto CUDA (*Compute Unified Device Architecture*).

Si tratta essenzialmente di un linguaggio di programmazione simile al C, con possibilità di essere integrato sia con C/C++ che Python. Con CUDA si possono scrivere programmi che sfruttino il parallelismo sui dati, ad esempio operazioni su grandi matrici o immagini, render di video in alta definizione ecc. Tanto più si riesce a massimizzare la quantità di queste operazioni che si svolgono nella GPU, tanto più si ha un guadagno in termini prestazionali.

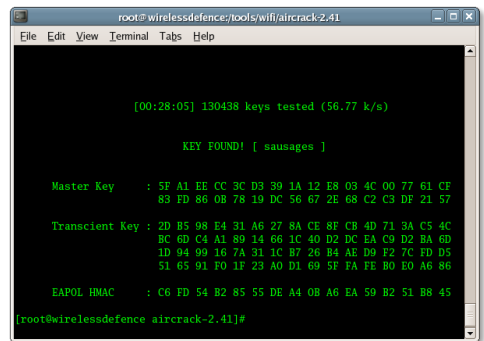
La fonte migliore di informazioni su CUDA è naturalmente il sito www.nvidia.com dove si possono trovare tutti i dettagli architetturali qui volutamente trascurati.

Aircrack + CUDA

Cosa c'entra il noto programma per testare la sicurezza delle reti wireless (serve a questo :P) con CUDA? A qualcuno è venuto in mente di velocizzare l'operazione di cracking della password di una rete wireless sfruttando CUDA, in quanto provare una password è proprio il genere di operazione ripetitiva su dati diversi che può sfruttare una GPU.

Recuperati i sorgenti di Aircrack, è stato necessario individuare le funzioni che si occupavano di testare la correttezza di una chiave WPA-PSK e scriverne una versione CUDA, da chiamare al posto di queste qualora fosse disponibile sul sistema una scheda video supportata.

Il risultato dell'operazione non è straordinario ma è significativo: una scheda video come la nVidia 9800GT può testare circa tre chiavi e mezzo nello stesso tempo in cui un processore quad core a 2,33GHz ne testa una. E questo senza particolari ottimizzazioni, modificando un codice già esistente piuttosto che scrivendone uno ad hoc.



```

root@wirelessdefence/tools/wifi/aircrack2.41
File Edit View Terminal Tabs Help

[00:28:05] 130438 keys tested (56.77 k/s)

KEY FOUND! [ sausages ]

Master Key   : 5F A1 EE CC 3C D3 39 1A 12 E8 03 4C 00 77 61 CF
              83 FD 86 0B 78 19 DC 56 07 2E 68 C2 C3 DF 21 57

Transient Key: 2D B5 98 E4 31 A6 27 8A CE 8F C9 4D 71 9A C5 4C
              8C 6D C4 A1 89 14 66 1C 4D D2 DC EA C9 D2 BA 6D
              1D 94 99 16 7A 31 1C B7 26 B4 AB D9 F2 7C FD D5
              51 65 91 F0 1F 23 A0 D1 69 5F FA FE B0 E0 A6 86

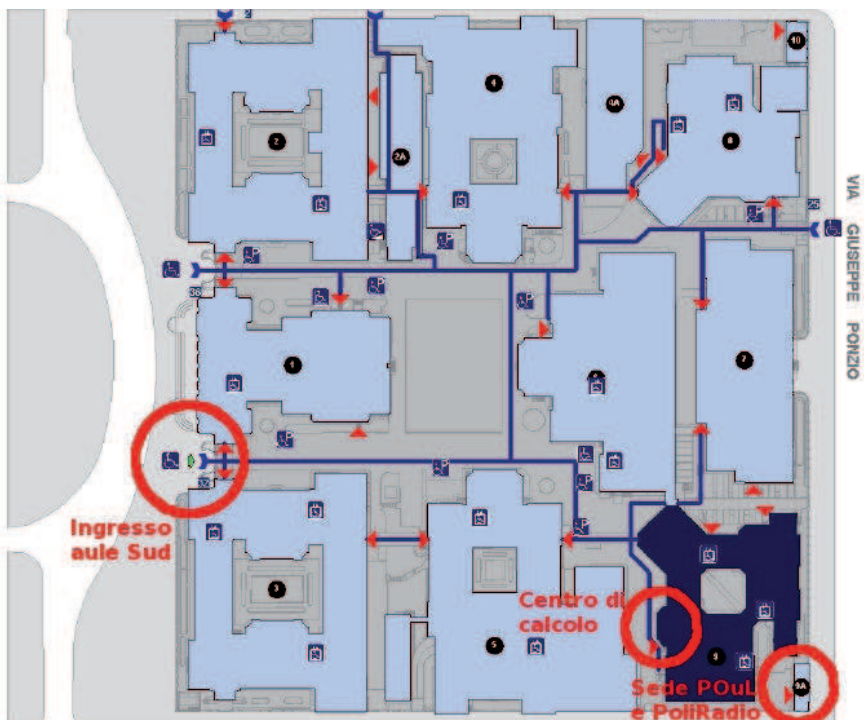
EAPOL HMAC  : C6 FD 54 B2 85 55 DE A4 0B A6 EA 59 B2 51 B8 45

[root@wirelessdefence aircrack-2.41]#

```

I limiti di CUDA sono comunque evidenti, dovuti principalmente alla sua esclusività per le schede nVidia. Per questo è stato creato anche OpenCL (*Open Computing Language*) che unifica il linguaggio di programmazione e permette di eseguire il codice sia su CPU che GPU, sia per schede nVidia che ATI.

Con questi presupposti, siamo fiduciosi che in futuro potremo spremere sempre più potenza dal nostro hardware, e che dovremo scegliere password sempre più difficili per le nostre reti wireless ;D



Vi è venuta voglia di conoscere il mondo di Linux? Volete partecipare più da vicino alle nostre attività? Volete scrivere un articolo su questa rivista? Iscrivetevi alla nostra mailing list oppure venite a trovarci presso la nostra sede!

sito Internet: www.poul.org
informazioni: info@poul.org



La stampa della rivista è interamente finanziata dal Politecnico di Milano, che non si assume alcuna responsabilità sul contenuto.

Stampa a cura di *Acheias* di S. Siragusa, Milano 2009.