Corso di Programmazione Concorrente

Tesine 2016-2017

Valter Crescenzi

http://crescenzi.inf.uniroma3.it

Tesine vs Homework

- Gli homework rappresentano un percorso di studio guidato
  - La soluzione è già ampiamente delineata nel testo
- Le tesine rappresentano un momento di studio “libero” e non vincolato, a problemi non banali e di chiarissima utilità pratica contingente
- Ma *ostinatamente* allineati agli obiettivi formativi di questo corso sulla programmazione concorrente

Sommario

- Tesine (Max 1-2-3 persone per tesina>>)
  - Speed-up Contests
    - XPath Data Extraction
    - XPath Fixed Point
  - Lettura di articoli scientifici
    - Su algoritmi *non-bloccanti*
    - Swarm Locking
    - Biased Locking
  - Crawling
    - Generico: BubiNG
    - *Parallel Corpora (DiffWeb)*
  - Akka
    - Risolvere HWC2 con Attori (Homework *HWA*)
  - Hypertextual Logging

Speed-up Contest: Tesina *XPath*

- Max 2 persone
- Si vuole parallelizzare un algoritmo per il calcolo massivo di espressioni XPath su collezioni di pagine Web rispondenti al medesimo HTML template
- Obiettivo: massimo speed-up
- Valutatore di N~10^3 XPath su M~10^3 pagine
- Utilizzando Cyberneko per caricare i DOM di pagine Web
  - *http://nekohtml.sourceforge.net/*
- A sua volta questo utilizza una libreria nota come Xerces
  - *https://xerces.apache.org/xerces2-j/faq-dom.html*
- Che però non è thread safe!
  - *https://xerces.apache.org/xerces2-j/faq-dom.html#faq-1*

Tesina *BubiNG*

- Max 3 persone
- Il crawling è uno dei problemi più *oscurato* dall'affermazione delle grosse tech-company
- Un crawler sviluppato da un gruppo di ricerca di Milano ma che ambisce ad essere competitivo con quello sviluppato dalle grosse company
- Descritto in un articolo scientifico:
  - *https://arxiv.org/pdf/1601.06919v1.pdf*
- Open Source!
  - *http://law.di.unimi.it/software/*
- Tesina in diverse fasi
  1) Studio articolo
  2) Studio codice
  3) Presentazione
  4) Esecuzioni codice
  5) [Customizzazioni]

Speed-up Contest: Tesina *FixedPoint*

- Max 1 persona
- Un algoritmo innovativo per l'inferenza automatica di espressioni XPath
- Data una collezione di pagine $\mathcal{P}$ che rispondono allo stesso HTML template
- Max speed-up per un algoritmo che itera queste fasi:
  - Generazione di un insieme di regole XPath che estraggono tutti i valori di un campione di pagine
  1) Estrazione di vettori di valori tramite le regole
  2) Rigenerazione delle regole per i valori estratti
  - Ripeti 1-2 sino alla convergenza (dell'ins. regole/vettori/valori estratti)

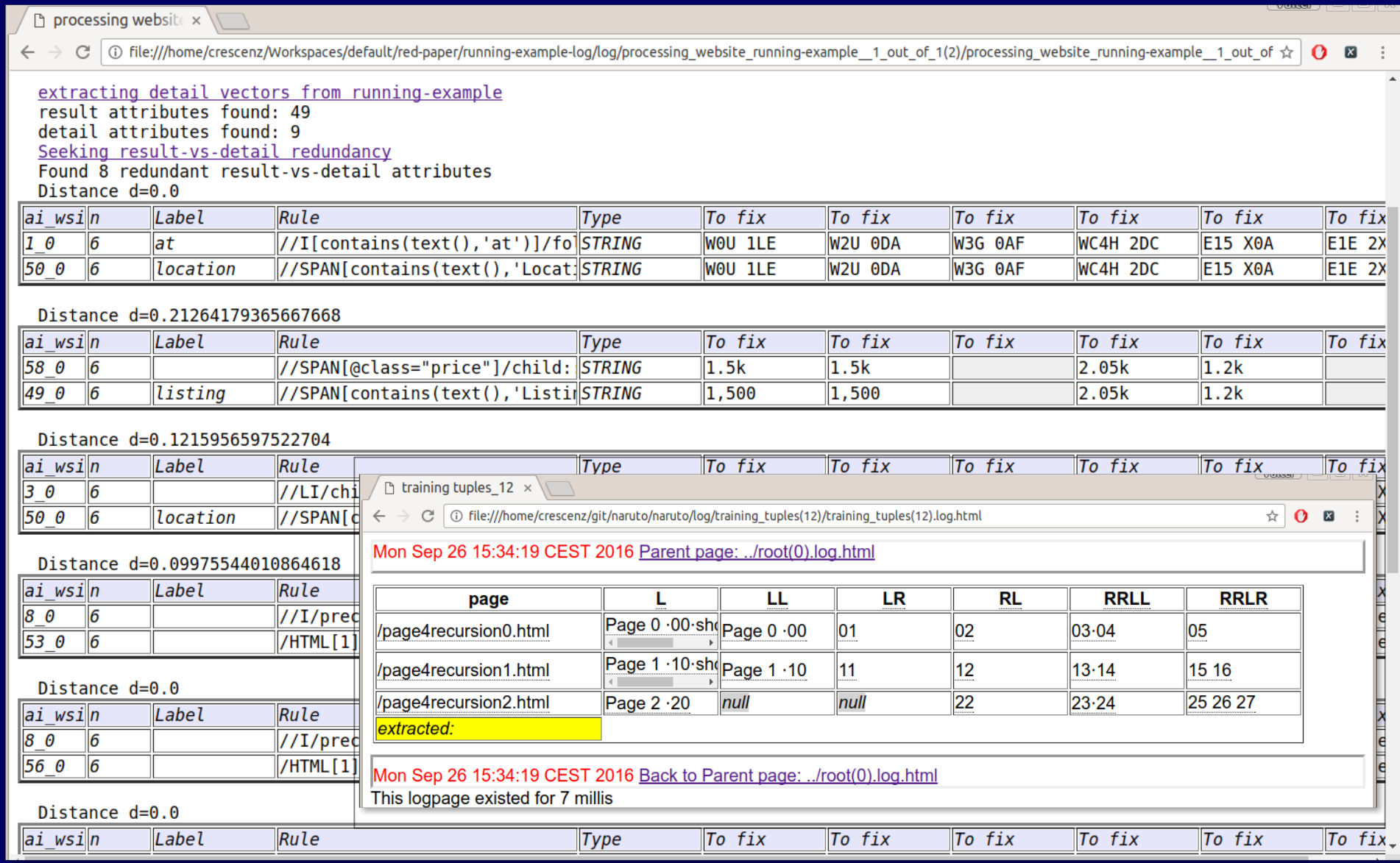
Tesina *Swarm* (max 2 persone)

- Max 2 persone
- Le grosse collezioni (dinamiche) non vengono mai protette da un solo lock
- *Lock-splitting:* molteplici lock per proteggere *regioni* diverse della collezione.
- Al crescere del numero di lock:
  - Migliora il livello di parallelismo consentito
  - Peggiora l'uso di memoria
- Alberi, Grafi, ed in generale enormi collezioni dinamiche possono crescere in direzioni non prevedibili creando squilibri in ogni distribuzione inizialmente equa dei lock
- Vediamo i lock come membri di uno swarn per "proteggere" l'intera collezione
- Gli accessi concorrenti possono perturbare l'equilibrio:
  - più si concentrano nel tempo e nello spazio, più dovrebbe essere aumentata la granularità, per non minare i livello di concorrenza
  - più si dirano nel tempo e nello spazio più i lock, più dovrebbero rimuoversi perché sostanzialmente inutili
- I lock possono:
  - *muovere* verso le regioni sino a raggiungere un punto di equilibrio
  - *sdoppiarsi/morire* in funzione della dimensione e forma della struttura

Tesina *HLog*

- Max 2 persone
- La consultazione e di log è un'attività
  - molto onerosa
  - Log di codice multi-thread spesso illegibili
- HLog: un framework per il log ipertestuale
  - Il log è organizzato in pagine HTML
  - Le pagine formano un sito intero
  - Le chiamate di metodi possono facilmente essere loggate con link pagina padre/figlia
  - Diversi thread possono loggare in distinte pagine/percorsi del sito di log
- Specializzare HLog per il supporto del logging di codice multi-thread

Esempio di Log Ipertestuale



processing website

file:///home/crescenz/Workspaces/default/red-paper/running-example-log/log/processing_website_running-example__1_out_of_1(2)/processing_website_running-example__1_out_of

extracting detail vectors from running-example
result attributes found: 49
detail attributes found: 9
Seeking result-vs-detail redundancy
Found 8 redundant result-vs-detail attributes
Distance d=0.0

ai_wsi	n	Label	Rule	Type	To fix	To fix	To fix	To fix	To fix	To fix
1_0	6	at	//I[contains(text(),'at')]/fo	STRING	W0U 1LE	W2U 0DA	W3G 0AF	WC4H 2DC	E15 X0A	E1E 2X
50_0	6	location	//SPAN[contains(text(),'Locat	STRING	W0U 1LE	W2U 0DA	W3G 0AF	WC4H 2DC	E15 X0A	E1E 2X

Distance d=0.21264179365667668

ai_wsi	n	Label	Rule	Type	To fix	To fix	To fix	To fix	To fix	To fix
58_0	6		//SPAN[@class="price"]/child:	STRING	1.5k	1.5k		2.05k	1.2k	
49_0	6	listing	//SPAN[contains(text(),'Listi	STRING	1,500	1,500		2.05k	1.2k	

Distance d=0.1215956597522704

ai_wsi	n	Label	Rule	Type	To fix	To fix	To fix	To fix	To fix	To fix
3_0	6		//LI/chi							
50_0	6	location	//SPAN[c							

Distance d=0.09975544010864618

ai_wsi	n	Label	Rule
8_0	6		//I/prec
53_0	6		/HTML[1]

Distance d=0.0

ai_wsi	n	Label	Rule
8_0	6		//I/prec
56_0	6		/HTML[1]

Distance d=0.0

ai_wsi	n	Label	Rule	Type	To fix	To fix	To fix	To fix	To fix	To fix

training tuples_12

file:///home/crescenz/git/naruto/naruto/log/training_tuples(12)/training_tuples(12).log.html

Mon Sep 26 15:34:19 CEST 2016 Parent page: ../root(0).log.html

page	L	LL	LR	RL	RRLL	RRLR
/page4recursion0.html	Page 0 ·00·sh	Page 0 ·00	01	02	03·04	05
/page4recursion1.html	Page 1 ·10·sh	Page 1 ·10	11	12	13·14	15 16
/page4recursion2.html	Page 2 ·20	null	null	22	23·24	25 26 27
extracted:						

Mon Sep 26 15:34:19 CEST 2016 Back to Parent page: ../root(0).log.html
This logpage existed for 7 millis

Tesina *JACM* (max 1 persona)

- Leggere, commentare, (e farmi capire!) uno tra gli articoli scientifici scelto tra i seguenti, eventualmente preparando una breve presentazione
  - *Split-ordered lists: Lock-free extensible hash tables*
    Ori Shalev, Nir Shavit

    http://dl.acm.org/citation.cfmdoid=1147954.1147958
  - *DomLock: a new multi-granularity locking technique for hierarchies.*

    Saurabh Kalikar, Rupesh Nasre

    http://dl.acm.org/citation.cfm?id=2851164
  - *Fine-grained adaptive biased locking*

    Filip Pizlo, Daniel Frampton, Antony L. Hosking

    http://dl.acm.org/citation.cfm?id=2093157.2093184

Tesina *DiffWeb* (max 2 persone)

- E' stato ideato un algoritmo di crawling innovativo specializzato per trovare e catturare parallel-corpora da siti web strutturati e multi-lingua

- Parallel corpora: collezione di *documenti* di medesimo contenuto tradotti in più lingue

ZH	天下大勢，分久必合，合久必分。
EN	Long divided, the world will unite; long united, it will fall apart.
PT	O império, unir-se-á após a divisão e dividir-se-á após a união

Perché Accumulare Parallel Corpora?

- Motivazioni
  - Esiste una fiorente industria, tuttora in forte espansione, per la traduzione professionale

- Esistono diversi algoritmi pubblici ML per la MT
  - Machine-Translation
- Si possono scaricare gratuitamente da internet, non sono considerati un asset da proteggere! Perché?
- Perché si tratta di algoritmi *statistici* **affamati** di **enormi** quantità di esempi
- Il vero asset sono i *training data*!
  - che difatti nessuno pubblica...

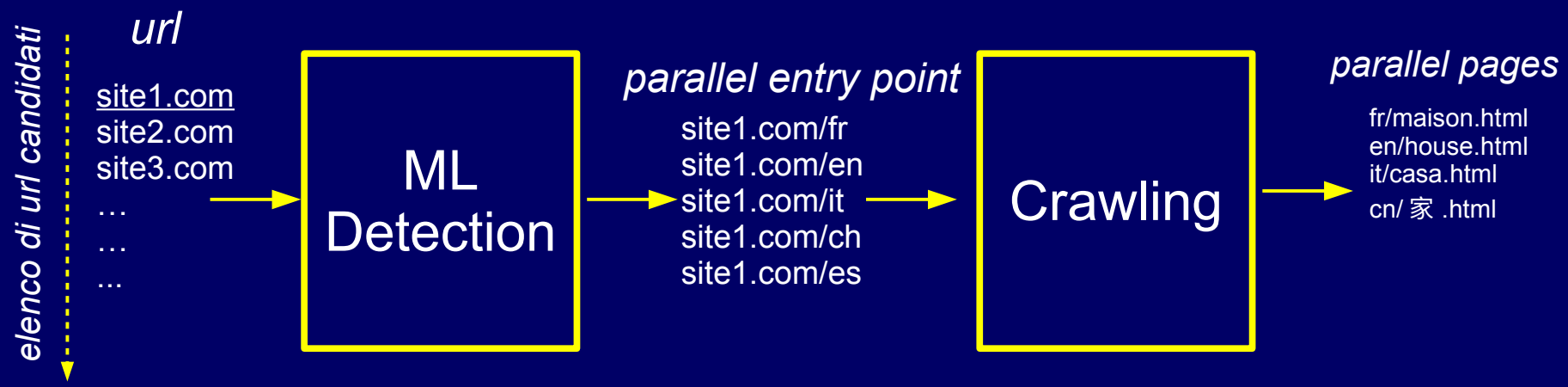
Crawling di Parallel Corpora (1)

- Non è un dettaglio: per essere competitivi c'è bisogno di esempi per *miliardi di parole*
- Google-Translate si basa su un *crawling esaustivo del Web*
  - in presenza di risorse di calcolo praticamente illimitate e dello snapshot del Web…

- E' possibile collezionare i siti che offrono i parallel-corpora anche senza essere Google
  - Basta ultra-specializzare gli algoritmi di crawling
  - Crawling efficiente e mirato della *sola* porzione di interesse del Web

Crawling di Parallel Corpora (2)

- Consiste di due fasi
  - *Detection:* “scovare” i siti multi-lingua e paralleli
  - *Crawling:* visitare in parallelo le sezioni di ogni lingua



Tesina *DiffWeb* (max 2 persone)

- Progettare un'architettura ad attori dell'intero sistema
  - ovvero definire quali siano gli attori principali e cosa comunicano
- Sviluppare alcuni di questi attori>>

Tesina *MLDetect* (max 2 persone)

- E' un modulo che si occupa di decidere se un sito è effettivamente parallelo (ovvero multi-lingua e con contenuti paralleli, e regolarmente strutturato)
  - N.B. le nostre migliori stime dicono che solo 1-4% dei siti lo siano!
- Si basa su varie euristiche, talune onerose
- Va progettato secondo un modello ad attori con questi requisiti:
  - bisogna processare il 96-99% dei siti che NON sono paralleli molto velocemente
  - Perché costano x100 !!!
- N.B. esistono note librerie esterne che permetto la language-detection in maniera affidabile ed efficiente

Tesina *DimIO (max 2 persone)*

- Dimensionamento del corretto numero di f.d.e./attori delle componenti I/O-intensive nell'architettura di cui prima
- I principali motivi di I/O sono:
  - fetching delle pagine
  - caching delle pagine

Tesina *ParallEP* (max 2 persone)

- Sviluppare un algoritmo concorrente efficiente per trovare entry point paralleli che non siano banalmente le homepage
- L'assunzione attuale è che molti siti multilingua strutturati siano paralleli già dalla home page. Purtroppo non è sempre vera:
  - i portali per l'intermediazione nel settore alberghiero quasi sempre nascondono i contenuti “veri” dietro un primo strato molto “localizzato e customizzato” per ciascun paese in cui operano
  - Bisogna saper navigare particolari tipi di
    - *Form*
    - *Link con bandierine*

Tesina *Stream* (max 2 persone)

- La fase di detection e di crawling consente di dimensionare a ragion veduta la percentuale di risorse allocate per le due fasi messe in pipelining
  - Il crawling parallelo permette di ottenere molti testi-paralleli da siti più produttivi scaricando meno pagine (ed abbandonando i percorsi di navigazione meno promettenti)
  - Ma necessita di una più ampia scelta di siti per dare frutti: deve poter trovare percorsi di crawling migliori
  - Cercare siti e non usarli costa!!! (vedi x100 di prima)
  - C'è un trade-off da risolvere….
  - ✔ Ideare una soluzione utilizzando gli *Akka Stream*
    - Permettono di costruire pipeline di processamento basandosi sulla back-pressure

Tesina *HWA* (max 1 persona)

- Risolvere HWC2_THREAD con Attori
  - Attenzione: bisogna rispettare i limiti imposti dal modello ad attori (ad es. msg *immutabili*)
- Confrontare le prestazioni vs HWC2_THREAD
  - Misurando lo speed-up sulla stessa architettura multi-core delle due soluzioni