Akka Homework - Task 2

Isabel Bär, Yannik Schröder

Distributed Data Management SS 2020

Agenda

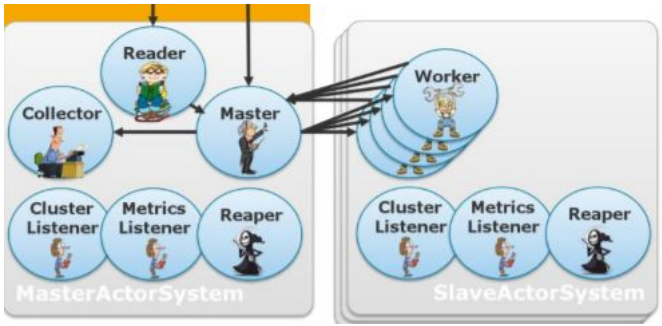
1. **Considerations**

a) Task and Architecture
b) Considerations On Cracking Passwords
c) Considerations On Parallelization

2. Overview of Our Cornerstones

a) Setting The Threshold-Based Strategy
b) Pull Propagation
c) Iterators

Task and Architecture



Homework

Task 3 – Password Cracking

	Name	PasswordChars	PasswordLength	Password	Hint1	Hint2	Hint3	Hint4	Hint5	Hint6	Hint7
	Sophia	ABCDEFGHIJK	10	GGGFGFFFFG	HJKGDEFBIC	FCJADEKGHI	FAJBDIEKGH	AGCJEHFKIB	BHKICGFADJ	JIFAGKDBCE	GAHDKJBCEF
	Jackson	ABCDEFGHIJK	10	EFFF		AEHJIDGFKC	IDAHFGEKBJ	EHFIJKBGAC	HFJIEDACBK	FGKIDJCEAB	KDHGCAEJFB
	Olivia	ABCDEFGHIJK	10	KDDI		CAKEIFHGJD	JBFEDHIKAG	IDAKGHBFJC	KGBAEICHDJ	DKHFBEJIAC	EABJGFIKDC
	Liam	ABCDEFGHIJK	10	CCCGGGC		FAICGJDHEK	CHBKIGEJAF	AICDKGHJBF	EDAGKBJHIC	JDKIFACEGB	BGKJDAHCFE
	Emma	ABCDEFGHIJK	10	BDDBDDDDDB		HEAJIBDGFK	BAHCKDFIJG	HBEDKAGCIJ	IBHCEFJADK	FAGDEJICKB	GFHEAKCDBJ
	Noah	ABCDEFGHIJK	10	GHGGHGGHHH		CFKBJGDIEH	CAIGHEJFDK	GJBEKIADFH	AIBCJHGEKF	GDIBCKFHAJ	CGJHDEAIBK	DGKFBEACJH
	Ava	ABCDEFGHIJK	10	DEEEDFDFDD		FHIKEBGDJC	KHFICAJGED	KIAHDFEJGB	CGFAKIBDHJ	ACEHFKBIDJ	GBADIJEKFC	AFCKGHBDJE
	Aiden	ABCDEFGHIJK	10	HHIHI		GCIFEHDKBJ	JDHIEGKACF	FJHBEGAKDI	AIBJEHKGFC	CGJAFBIHDK	JECAIDGHBK	IJBDCKEAFH
	Isabella	ABCDEFGHIJK	10	CJCJCCJ		EHDCGIKBJF	IFJCAEHGKD	AFBHEGKIJC	KGFBIADJHC	JKAGEDHIBC	CBKIDEAHFJ	CJAFKEIBDG
0	Lucas	ABCDEFGHIJK	10	BBCBCCC		KGJHIDECFB	BHFACKEGIJ	ICJGHFKBAD	KEICHGAJDB	BCDKEJIFAH	IDGEBAJKCF	FCBDKGHJAE
1	Mia	ABCDEFGHIJK	10	IDDDDDIDDI		FIBG						BKIGAEDFCJ
2	Caden	ABCDEFGHIJK	10	DDDAADDDDD		AEIC						EKFGAJCBHD
3	Aria	ABCDEFGHIJK	10	CCCFCCFFCC		GKH						HBJEAFDCGK
4	Grayson	ABCDEFGHIJK	10	JJBBJJ		GEH						BKGEJFACDI
5	Riley	ABCDEFGHIJK	10	BGGGBBB		GHE						JBCHAGEKDF
6	Mason	ABCDEFGHIJK	10	AAJAJA		EKJ						BCGEFAKJID
7	Zoe	ABCDEFGHIJK	10	JJJJJJ		JHC						CBEFIDGKAJ
8	Elijah	ABCDEFGHIJK	10	EJJEJEJJEE		KDC						JBCGADIEKF
9	Amelia	ABCDEFGHIJK	10	GDDGGGDGDG		GCI						HEGDCAKJFB
0	Logan	ABCDEFGHIJK	10	FFEFEEEFFF		KHB						JFHDCKEAGB
1	Layla	ABCDEFGHIJK	10	CCCHCHCCCC		GIF						JBGAHFDCEK
2	Oliver	ABCDEFGHIJK	10	ABBBAABAAA		AFK						FJGKHEADBC
3	Charlotte	ABCDEFGHIJK	10	BGBGBGBBBG		ECK						GJBECKAFHD
4	Ethan	ABCDEFGHIJK	10	HHBBHHH		DHJ						BCHJEAKFDG
5	Aubrey	ABCDEFGHIJK	10	EJJEEJE		HDFEJBKICG	IF	AIGEBIK	KCEFBAIGHJ	IDCJBAGEHK	IKHJCDBAEF	JCGIFDBEAK
6	Jayden	ABCDEFGHIJK	10	CCCGGGG		DJEHCBKFIC	AIDHE	ECFKBIAHJG	GDCIFKBJAH	HJBGAIKCED	DICKFBGAEJ	GFDCKBAHJE
7	Lily	ABCDEFGHIJK	10	DHHDHDD		KCJFIBHEDG	FJDKCAIEGH	DABGJEFKIH	DCGIKHFABJ	KDGBEHIACJ	IBCKDFHJAE	EHABDKCFJG
8	Muham	ABCDEFGHIJK	10	CBCBBBC		EDFGHKIBJC	HEICAKBJFG	CABIDFGHKJ	DHAKICBGJE	IHAKJCEBFD	CEABFJGKID	ABGKFDHCEJ
9	Chloe	ABCDEFGHIJK	10	CEECECCCEC		KEGDHFCBIJ	GCIHAEDKFJ	HFGKIBACEJ	CJHGKBDAIE	FECIBJKADH	GAFCIBEKJD	CJBAKEGDFH
0	Carter	ABCDEFGHIJK	10	BBIIBIBIIB		CKFGBIHDEJ	EJDKIHGABF	ECHIJGFBAK	AFHIBCKGDJ	IHCBKGEJAD	AFHIDJKBCE	IBGCJKFAED
1	Harper	ABCDEFGHIJK	10	EAAAEAEEAE		IAFKCHJGDE	AKFDJIHGBE	BGECFIJKAH	BJDAIGKEHC	IHEBKACJDF	BJDIGAKFCE	EFADJKCBGH
2	Michael	ABCDEFGHIJK	10	DCDDCCDDDD		JGFEICBKHD	CKJDHGIEAF	KIGDHABJCF	GDEIHACBJK	BICEAFDHKJ	JGFCKBDEAI	HCBGDKFAJE
3	Evelyn	ABCDEFGHIJK	10	IICICCICCC		CDBJHIEGKF	FIKGEHCAJD	BIJAGEKFHC	CADBIHFJKG	GHJDBKAEIC	KJEIDHABCF	ACBGKFDIJE
4	Sebastian	ABCDEFGHIJK	10	EIIIEIIIIE		EDFKHGJBIC	KFHIDJGACE	KHAIDGBJFE	CIABGJKEHF	JKEIDCBAHG	KIDFJABHEC	FDBGJCAEIK
5	Adalyn	ABCDEFGHIJK	10	JJJJDJDDJJ		IKEJFCBHDG	AHDCFKEJGI	DBIKAHJGEF	DKAIHGFJCB	CABKHEDJIG	ABJEFCHDKI	GJAFEBCDIK
6	Alexander	ABCDEFGHIJK	10	HKHKHKKHKK		CEBHKDFIJG	FGJADKCIEH	AGDIFHKJBE	FHCEKABGIJ	BFJAIDCGHK	BCAHEJDGIK	JFCDIHKAEB

Passwords to be cracked

All characters that may appear in the password

Number of characters in the password

These two fields have always the same value for all records.

Hints:
- Every hint contains all PasswordChars besides one char, i.e., |Hint|=|PasswordChars|-1
- The missing char is the hint, because it does not appear in the password.
- The number of hints can change!
- The more hints we have, the easier it is to find the password.

Considerations on Cracking Passwords

Cracking Hints:

- Generate all permutations of the given character universe
  - Leave out the last char of each permutation
  - That's the possible hint character

Cracking Passwords:

- Guess the password chars by generating all combinations of the possible characters without repetition
- The complexity of this task decreases with more hints!
- Guess the password by using the previously generated chars

- Complexity varies depending on password length, password characters, number of hints, etc.

  → **Need to dynamically decide how much hint cracking is optimal before password cracking**

Considerations on Parallelization

- Task Parallelism
  - Generation of workloads for hints and password
- Unit of parallelization
  - One entire password task/hint task
- Acceptable trade-off
  - Lots of passwords <-> difficult passwords

Alternative Architecture/Outlook:

- If passwords are really hard to crack
  - Partitioning of single tasks by dividing into ranges
  - Worker 1: AAAA-CCCB
  - Worker 2: CCCC-EEEE
- Spawn and destroy/stop actors working on the same problem when it is solved

Agenda

1. Considerations

a) Task and Architecture
b) Considerations On Cracking Passwords
c) Considerations On Parallelization

2. Overview of Our Cornerstones

a) Setting The Threshold-Based Strategy
b) Pull Propagation
c) Iterators

Overview Of Our Cornerstones

1. Calculation of the complexity of cracking a hint/password in order to find a threshold of how many hints should be cracked. The **threshold sets the pursued strategy** for work generation.

2. Implementation of a **Master-Worker-Pattern using Pull-Propagation** for task scheduling

3. **Usage of Iterators** for sequence generation

a) Setting The Threshold-Based Strategy

PasswordComplexity

Calculate the password complexity with hints and without hints and check:

if passwordComplexityWithHints < hintComplexity

If cracking the password directly is easier than cracking hints, no work for hint cracking is generated.

The master uses this threshold in the following methods:

- For a processed batch (processBatch), the master checks if it is necessary to crack hints (createHintWorkload)) or if the password can be brute-forced directly (createPasswordWorkload)

- After successfully cracking one hint, the master checks if more hints need to be cracked (handle HintSuccessMessage) or if a password workload can be generated

b) Pull-Propagation

Consumers ask the master for more work if they are ready. The work is stored in the queue of the master's state, which ensures that workers receive work accordingly to their capacity. Though pull-propagation generally leads to slower work propagation, the advantage overweighs because the risk of message congestion is smaller.

In our project, slower work propagation is not problematic as work is being propagated not very frequently. Thus, the potential increase of message overload can be neglected and implementing pull-propagation brings the advantage of avoiding message congestions without considerably decreasing the speed of work propagation.

c) Iterators

Implementation of **three Iterators** for:

- Permutations for hints
- Combinations of chars (which depends on the permutation for hints and changes accordingly)
- Combinations for one password,

We achieve the following advantages:

- Sequences can be generated on the fly (no need to keep all possible combinations in memory at all times)
- Waste of work can be minimized as the iteration immediately stops in response to success