6. domácí úloha: SWIM Failure Detector

Představte si, že jste se rozhodli si založit firmu. Váš předmět podnikání naneštěstí vyžaduje ukládat a zpracovávat data v řádech TB. Uložená data by navíc drtivou většinu času měla být dostupná. Tomuto požadavku vyhovíte s jedním serverem velice těžko, proto jste se rozhodli si vybudovat distribuovanou databázi bežící na komoditním hardwaru v podobě několika desítek serverů. V reálném světě bohužel nelze očekávat, že hardware bude mít vždy 100% spolehlivost, proto je třeba nějakým způsobem detekovat, zda nám nějaký proces náhodou neselhal.

S jedním způsobem detekce - all-to-all heartbeating jste se měli možnost seznámit na cvičení. Tento způsob detekce předpokládá velmi idealizovaný scénář, v němž se neztrácejí zprávy, a zpráv můžete odeslat kolik chcete. V tomto domácím úkolu však musíte počítat i s těmito omezeními, proto budete **implementovat protokol SWIM**, který znáte z přednášek a který by si už měl poradit i s takovýmto scénářem.

V našem případě si klidně můžete představit systém jako distribuovanou databázi - složenou z procesů - databází, u kterých je velká šance, že časem některý proces "umře". Ve vašem systému máte navíc jeden bezchybný proces v podobě "DisseminationProcess", u kterého máte zaručeno, že nikdy nespadne a zprávy, ve kterých je jako příjemce nebo odesílatel se nikdy neztratí. Tomuto procesu reportujete detekované procesy, které považujete za ukončené, aby tento proces mohl ostatním procesům sdělit, že reportovaný proces je mrtvý. V praxi by reportování "mrtvého" procesu mohl být například signál pro správce systému, aby šel ověřit nastalou situaci.

Z přednášek si jistě vybavíte vlastnosti detektorů. Jelikož se v našem příkladu snažíme co nejvíce přiblížit reálnému světu, budeme Vaše řešení i podle požadavků reálného světa evaluovat. To znamená, že by vaše řešení

1. *nemělo zbytečně vytěžovat síť,* aby nezpomalovalo samotnou práci databáze,
2. *mělo by být schopné detekovat všechny "mrtvé" procesy s rozumnou rychlostí,* aby se nám neztratila data, případně abychom to byli schopni rychle detekovat a
3. *mělo by být dostatečně přesné,* jelikož správce nemá čas běhat od serveru k serveru a zjišťovat, zda se jedná o falešný poplach.

Stáhněte si balíček hw07_detect.zip [https://cw.fel.cvut.cz/wiki/_media/courses/b4b36pdv/tutorials/hw07_detect.zip]. Vaším úkolem je doimplementovat třídu `ActStrategy` s metodou `act()`, kterou volá instance třídy `FailureDetectorProcess`. Metoda `act()` by měla implementovat protokol SWIM. V třídě `ActStrategy` můžete využít vlastní datové struktury, můžete také definovat vlastní třídy zpráv. V kódu k úloze najdete další nápovědu a vysvětlení dílčích částí kódu.

Zazipované soubory `ActStrategy.java` s případnými vlastními třídami (nesmí přepisovat kód projektu) odevzdávejte do systému BRUTE do **4. 5. 23:59 CET pro středeční cvičení** a **5. 5. 23:59 CET pro čtvrteční cvičení**. Za úlohu můžete získat maximálně **2 body** v závislosti na kvalitě vašeho řešení. Před odevzdáním se ujistěte, že jste z kódu odebrali všechny ladící výpisy.

Pokud využíváte vlastní třídy, musí se tyto třídy nacházet v package `swim` !

courses/b4b36pdv/tutorials/hw_06.txt · Last modified: 2023/03/02 00:56 by macejp1