A SALEVE RENDSZER

Dóbé Péter, dobe@iit.bme.hu
Molnár Zsolt, zsolt@zsoltmolnar.hu
Dr. Szeberényi Imre, szebi@iit.bme.hu
BME-IIT – BME-IK

Az ún. paraméterelemző feladatok olyan, a gyakorlati életben is sok helyen előforduló problémák, amelyek jellegükből adódóan nagyon egyszerűen szétoszthatók párhuzamosan végezhető részfeladatokra. Az alábbiakban bemutatunk egy szoftverfejlesztést segítő eszközt, amellyel ebbe a problémakörbe tartozó feladatot megoldó, párhuzamos számítási kapacitást is igénybevevő programok készíthetők.

1. Paraméterelemző feladatok a gyakorlatban

Ha ugyanazon számítási műveletsort egy nagy paramétertartományon hajtjuk végre, *paraméterelemző* (parameter study, PS) feladatról beszélünk. A különböző paraméterértékekre egymástól függetlenül végezhető a művelet, melynek következményeként a tartományt kisebb résztartományokra felosztva a számítás ezeken a résztartományokon konkurrensen, akár egyidőben futhat. Így ilyen feladatok esetében az elosztott számítási erőforrásokat könnyen ki lehet használni.

Számos helyen felbukkannak ilyen problémák a gyakorlatban, példaként említhetjük a részecskefizikai számításokat, ahol egy nem analitikus függvényt kell sok ezer helyen kiértékelni annak integráljának numerikus közelítéséhez. Emellett a statika, szilárdságtan témakörében is vannak példák ilyen számításokra, és sok egyéb fizikai szimuláció is ide tartozik.

Ez a széles körű megjelenés indokolja egy paraméterelemző feladatokat megkönnyítő segédeszköz szükségességét. Jelenleg ha egy kutató fel akarja használni a rendelkezésére álló párhuzamos számítási kapacitást, a számítógépére sokféle szoftvereszközt kell telepítenie, valamint meg kell tanulnia különböző, egymással

inkompatibilis hosszú távú ütemezők illetve Grid köztesrétegek kezelését. A Grid fejlesztések egyelőre kezdeti stádiumban vannak, ezzel együtt a megvalósítások egységességére törekvő szabványosítások is, melyek nem a grides alkalmazás készítését segítő fejlesztői környezetre, hanem inkább a Grid komponensek közötti mélyebb szintű együttműködésre irányulnak (például webszolgáltatások). További nehézség, hogy a programozó számára elsajátítandó technológiák többnyire idegenek a széles körben használt programozási nyelvektől.

2. A Saleve használata

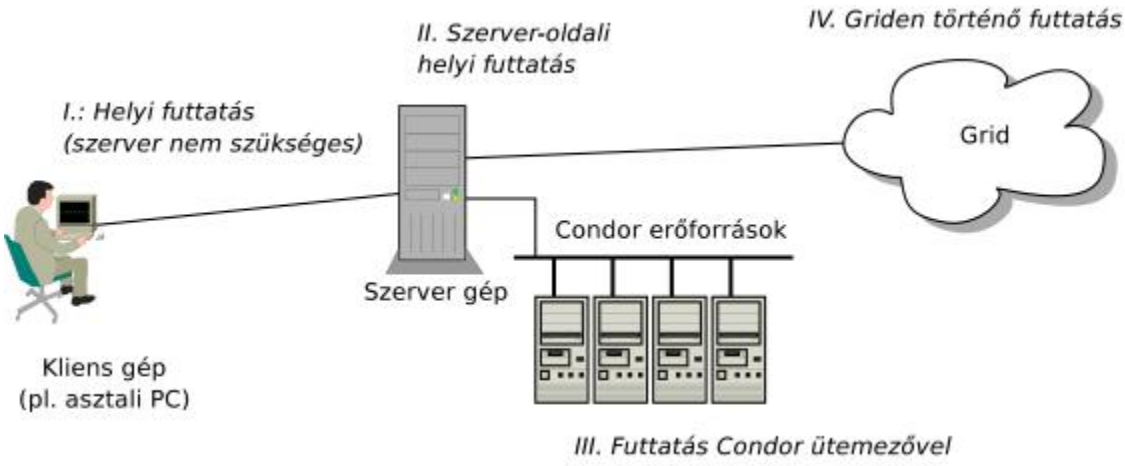
A paraméterelemző feladatok végrehajtásának egyszerűbbé tétele a *Saleve* projekt [1] célja. A Saleve egy nyílt forráskódú eszközkészlet, amely egy általános, elosztott technológiáktól független módszert nyújt C vagy C++ nyelven írt, párhuzamosan futni képes programok fejlesztésére. Használata igen egyszerű, hiszen már meglévő, szekvenciálisan futó programunkat forrás szinten csak kis mértékben szükséges megváltoztatni.

A módosított programkódot lefordítjuk és összeszerkesztjük a Saleve függvénykönyvtárral, így megkapjuk a Saleve kliens programot, amely többféleképpen képes futni. Tesztelési célra használhatjuk egy teljes értékű helyi lefutású programként, amely azonban, amennyiben több processzor áll rendelkezésre a futtató számítógépben, kihasználja ezt az adottságot, és szétosztja a feladatot részfeladatokra a processzorok között.

Másfelől viszont a programnak rendelkezésére bocsáthatjuk egy ún. *Saleve szerver* elérhetőségét egy környezeti változóban megadott URL formájában. Ekkor a kliens elküldi a saját bináris kódját és a bemeneti állományokat a Saleve szervernek. Innentől kezdve a szerver gondoskodik a feladat szétosztásáról, az eredmény begyűjtéséről és visszajuttatásáról a kliensnek. A kliens ilyenkor nem végzi el az egyes résztartományokon a számításokat, hanem a szervertől várja az ezeknek megfelelő részeredményeket, amelyek segítségével előállítja a végeredményt. Amennyiben hosszú idő múlva érkeznének meg a részeredmények, a várakozó kliens programot

megállíthatjuk, és egy feladat-azonosító megadásával később, akár egy másik gépen is folytathatjuk a várakozást a kimeneti adatokra.

Az, hogy a számítást végző program futhasson a távoli, idegen környezetben is, jelenleg csak úgy oldható meg, hogy a kliens programot statikusan szerkesztjük össze a Saleve könyvtárral és a számításhoz szükséges további függvénykönyvtárakkal.



1. ábra: a Saleve használatának esetei

Szerver használata esetén is több lehetőség van (lásd 1. ábra). Egyrészt futhat a feladat helyben a szerver gépen, amely előnyös lehet akkor, ha ez a gép erősebb számítási erőforrásokkal (például több processzorral) bír, mint a kliens PC. Másrészt viszont, ha rendelkezésünkre áll a háttérben egy többgépes klaszter (például *Condor* ütemezővel), vagy hozzáférésünk van egy Grid rendszerhez, a szerver ezt is ki tudja használni, amennyiben ezek interfészéhez van szerver-oldali támogatás egy plugin formájában (lásd később). Ilyen esetben a kliens számára teljesen közönbös, hogy a háttérben milyen elosztott számítási infrastruktúra található.

3. A moduláris Saleve szerver

A Saleve szerver feladata elsődlegesen a számítás és a bemenő adatok szétosztása számítási erőforrások között, valamint az elkészült részeredmények begyűjtése és egyesítése. A szerver felépítése lehetővé teszi a könnyű bővíthetőséget: ha egy új elosztott technológia használatához kívánunk támogatást nyújtani, elég egyszerűen egy

plugin-t készítenünk. A plugin fejlesztése annyiból áll, hogy néhány függvényt kell megvalósítani C++ nyelven: ezek a függvények a feladat beküldését és futási állapotuk ellenőrzését végzik. Az alapszerverben a helyi végrehajtás mellett támogatott a *Condor* [2], amely nagyteljesítményű számítások céljára egy igen elterjedt hosszútávú ütemező.

Fejlesztés alatt áll ezen kívül a gLite-nak [3], az Enabling Grids for E-sciencE (EGEE) [4] köztesrétegének az illesztése is a rendszerhez. Fontosnak tartottuk az EGEE Gridhez is fejleszteni támogatást, mivel ez Európa legjelentősebb Grid projektjei közé tartozik, 32 ország és közel 30 ezer processzor részvételével, így várhatóan lesz igény az EGEE használatára a Saleve felhasználói között is.

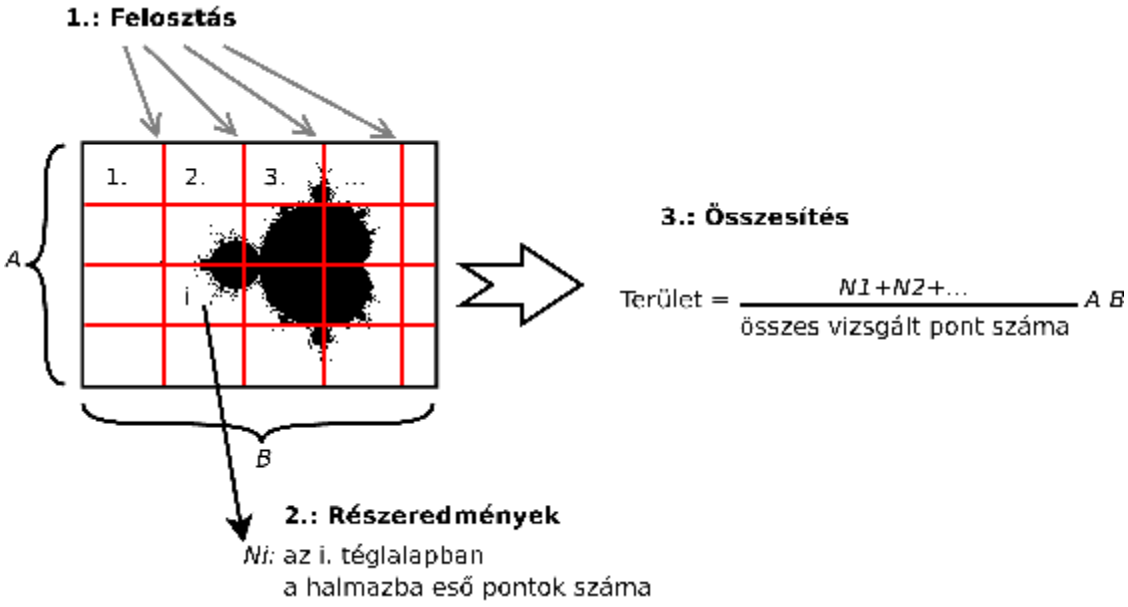
További fontos szerepe a szervernek a kliensek hitelesítése. Ez jelenti egyrészt a köztesréteg vagy ütemező felé történő hitelesítést, másrészt a szerver saját biztonsági elveinek megfelelő megszorításokat a kliensekkel szemben. Az előbbi függ a konkrét köztesréteg illetve ütemező technológiától, korszerű Grid esetén azonban általánosan elmondható, hogy egy X.509 tanúsítvány alapú hitelesítés az elterjedt módszer erre. A jelenlegi megvalósításban a tanúsítvány és a megfelelő titkos kulcs a szerver oldalon van tárolva, így a kliens oldalon semmilyen változtatás nem szükséges, amikor egy új plugint kívánunk használni a szerver oldalon. Ugyanakkor a szervert üzemeltetőre hárul a megfelelő felhasználói tanúsítványok beillesztésérnek feladata, továbbá a titkos kulcsok távoli oldalon történő tárolása is felvet biztonsági problémákat. A kliensnek a szerver felé történő hitelesítése egy SSL feletti HTTP kapcsolat révén megoldható, egy kliens oldali tanúsítvány bemutatásával.

A kliens és szerver közötti adatcsere, amelybe többek között a kliens bináris és a feldolgozandó adatok továbbítása tartozik bele, webszolgáltatás alapú. Ez jelen esetben teljes állományokon működik, de később lehetővé válik adatfolyamok átvitele webszolgáltatás alapokon az ún. ***webstream*** segítségével, amely a C++-ban ismert ***stream*** osztályokkal megegyező módon használható.

4. Egy példa

A népszerű Mandelbrot-halmaz [5] területének nagy pontosságú közelítését könnyen párhuzamosíthatjuk Saleve segítségével. A közelítést itt úgy végezzük, hogy egy könnyen számítható területű, a halmazt magába foglaló síkidom (például téglalap) nagyon sok pontjára megvizsgáljuk, hogy mekkora arányuk eleme a Mandelbrot-halmaznak, majd ezt az arányt megszorozzuk a síkidom területével.

A sík egy pontjának a halmazba tartozását egyszerű de időigényes számolással dönthetjük el. Fontos viszont, hogy más pontok halmazba tartozását nem kell ismernünk hozzá, vagyis pontonként függetlenül állapíthatjuk meg azt. Így a paramétertartomány felosztható résztartományokra, amelyeken egy-egy számítási példány futhat (lásd 2. ábra).



2. ábra: a Mandelbrot-halmaz területének kiszámítása a paramétertartomány felosztásával

A területszámító alkalmazásunk az első fázisban ezt a felosztást végzi el. A paraméter itt kétdimenziós: a valós és képzetes tengely koordinátáiból áll, a résztartományok pedig legegyszerűbb esetben téglalap alakú területeknek felelnek meg.

A második fázis a résztartományokon pontonként történő kiértékelés. Ennek eredménye, vagyis egy részfeladat kimeneti adata az ebbe a téglalapba eső megtalált halmazpontok száma.

A harmadik fázis a részeredmények összesítése: a talált pontok összegét elosztjuk az összes bejárt pont számával, majd ezt megszorozzuk a teljes vizsgált tartomány területével, így megkapjuk a végeredményt.

5. Köszönetnyilvánítás

E munka részben a Nemzeti Kutatási és Technológiai Hivatal Pázmány Péter programjának (RET-06/2005) támogatásával jött létre. A szerzők szeretnék kifejezni a köszönetüket az Európai Unió által támogatott EGEE projektnek (EU INFSO-RI-031688), valamint az NKFP MEGA (2_009_04) projektnek.

Hivatkozások:

[1] Zs. Molnár, I. Szeberényi, *Saleve: simple web-services based environment for parameter study applications*, In *The 6th IEEE/ACM International Workshop on Grid Computing*, 2005.

[2] *Condor Project*, http://cs.wisc.edu/condor/

[3] *gLite: Lightweight Middleware for Grid Computing*, http://glite.web.cern.ch/glite/

[4] *Enabling Grids for E-sciencE*, http://www.eu-egee.org/

[5] Benoît Mandelbrot, *Fractal aspects of the iteration of* $z \rightarrow lz(1-z)$ *for complex* l, z, Annals NY Acad. Sci. **357**, 249/259