Rendszerintegráció és -felügyelet laboratórium (VIMIM309)

Üzleti folyamatok felügyelete

Mérési segédlet

Készítette: Szombath István

Utolsó módosítás: 2011. április 14.

Verzió: 1.0

Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

1 Bevezető

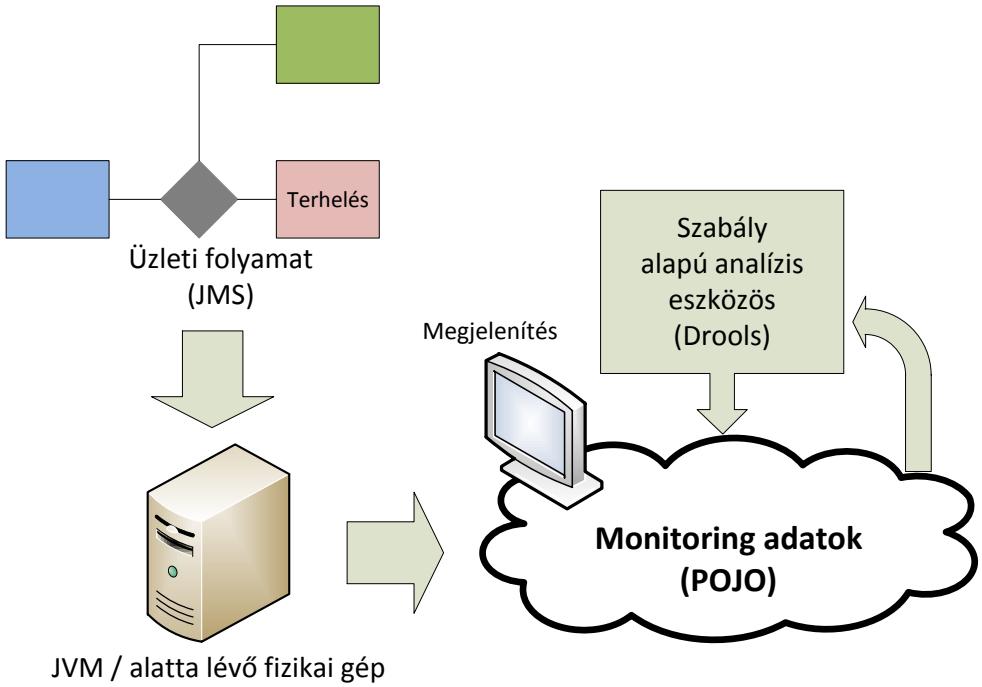
A labor során a méréseket végző hallgató a gyakorlatban is megismerkedik a rendszerintegráció és rendszerfelügyelet során használatos módszerekkel és eszközökkel. Végigköveti egy elosztott alkalmazás megvalósításának és felügyeletének legfontosabb lépéseit, ipari környezetben használt integrációs köztes réteg (middleware) technológiák és felügyeleti eszközök használatával. A mérések a következő témakörökhöz kapcsolódnak:

1. Munkafolyamatok megvalósítása Java nyelven
2. Megbízható üzenetküldés IBM WebSphere MQ alapon
3. Kommunikáció JMS és JMX technológia segítségével
4. OSGi szolgáltatások fejlesztése
5. Modell alapú eszközintegráció elosztott környezetben (SDE)
6. Szabályalapú üzleti logika
7. Üzleti folyamatok felügyelete

A megelőző méréseken a hallgatók különféle elosztott szolgáltatásintegráció megoldásokkal valósították meg egy mintarendszer üzleti logikáját. Jelen mérés során nem újabb üzleti logika implementáció kerül megvalósításra, hanem egy már meglévő üzleti logika szabály alapú felügyeletére kerül sor. A meglévő üzleti logika, mellyel dolgozni fogunk a mérés során, a „Kommunikáció JMS és JMX technológia segítségével” c. mérés (továbbiakban JMS mérés) során elkészített üzleti logika fogja szolgáltatni. Tehát szükség lesz a JMS mérés során elkészített forráskódra. Az üzleti logika felügyeletére szabály alapú módon kerül sor, azaz a „Szabályalapú üzleti logika” c. mérés során elsajátított, JBoss Drools Expert termék ismeretekre is szükség lesz.

A mérés során a hallgatók elsajátíthatják a szabály alapú felügyeleti rendszerek alapelveit, és megismerkedhetnek használatával.

2 A mérés vázlata



ábra 1 A mérés vázlata

A mérés során az előző méréseken megírt kódra és elsajátított ismeretekre lesz szükség. A feladat egy valós üzleti folyamat megvalósítása, felműszerezése, monitorozása, és az üzleti folyamat szabály alapú felügyelete:

1. A mérés előfeltétele az üzleti folyamat JMS implementációja. Az üzleti folyamat alapegységeit (taskjait) ki kell egészíteni, hogy valóban erőforrás intenzív feladatot lássanak el. Például végezzenek memória vagy CPU intenzív számítást. Gondoskodni kell arról is, hogy ezúttal nem csupán egy lefutást vizsgálunk, hanem több párhuzamos lefutást.
2. Ez után válasszuk ki a mérendő paramétereket, majd gondoskodjuk arról, hogy ezeket mérni is tudjuk. Egyrészt lehetőség van az üzleti folyamatot futtató kód felműszerezésére, ezzel mérhetünk magas,alkalmazás szintű metrikákat (kérés-áteresztőképesség számítása, átlagos feldolgozási idő, stb), másrészt a JVM, illetve a, JVM-et futtató gép/operációs rendszer paramétereit is figyelhetjük (ezzel alacsony szintű metrikákat célszerű mérni, pl CPU terhelést). Mindenképp mérjünk üzleti folyamat szintű, illetve alacsony szintű metrikákat is!
3. A mért adatokat eljuttatjuk a saját monitoring alkalmazásunkba, mely jelen esetben egy Drools Expert rendszer lesz. Tehát, gondoskodnunk kell:
    a. a felügyelni kívánt (monitorozandó) eszközök metamodelljéről (ábra + POJO implementáció),
    b. példánymodell létrehozásáról,
    c. mért adatok beszúrására a példánymodell megfelelő elemének megfelelő attribútumába (eseményvezérelten, vagy poll jelleggel).
4. Felügyeleti szabályokat hozunk létre, majd a rendszerünket futtatva figyeljük a lefutást, folyamatos terhelés alatt. Példák a felügyeleti szabályokra:
    a. riasztás, ha a processzorhasználat a hoszton >80%,
    b. riasztás, ha kevés a rendelkezésre álló szabad memória,
    c. riasztás, ha esik az áteresztőképesség, nő a válaszidő vagy éppen megugrik a kérések átlagos száma,
    d. ezek tetszőleges kombinációja, különösen érdekes a magas szintű és az alacsony szintű metrikák egymásra hatása (pl áteresztőképesség – CPU használat).
5. Végül ábrázoljuk az egyes erőforrások terheltségét grafikusan. Erre a JfreeChart könyvtárat ajánljuk. A cél, akár több idősor ábrázolása, mely az erőforrások kihasználtságát mutatja az idő függvényében. A metamodellben nem szükséges az erőforrások „history” jellegű kihasználtságának tárolása, de nem is tiltott.

3 Háttérismeretek

A mérés során szükség lesz a „Kommunikáció JMS és JMX technológia segítségével” valamint a „Szabályalapú üzleti logika” c. gyakorlat anyagának ismeretére.

3.1 Üzleti folyamat implementációja

A JMS mérés üzleti folyamat implementációjának terheléssel való felműszerezését a hallgatóra bízzuk. CPU és memória intenzív feladatokat ajánlunk, de megfelelő indoklással más is elfogadható, sőt…

Az üzleti folyamatot ezután futtatjuk, azaz folyamatosan kérésekkel terheljük az üzleti folyamatot. Ezt a kódot nyugodtan lehet a folyamatot indító kódba szőni.

3.2 Üzleti és alacsony szintű metrikák

A mérendő paraméterek tekintetében is szabad kezet kapnak a hallgatók, ehhez bemutatunk pár irányelvet:

Vannak paraméterek, melyek az adott folyamatra vagy az azt futtató környezetre vonatkoznak, vannak melyeket a JVM-ből ki lehet nyerni, megint másokat a JVM-et futtató gépből célszerű.

Példák:

Egy egyszerű kódrészlet, ami megmutatja, hogyan kell a futó JVM gép paramétereit kinyerni:

http://www.rune-server.org/programming/application-development/178277-cpu-memory-usage-statistics-java-application.html

Példakód arra, hogy egy JMS queue hosszát hogyan lehet lekérdezni:

http://community.jboss.org/thread/52523

A külső JVM-et futtató gép paramétereit is elérhetjük, erre egy példa:

Így kezdeményezhetünk a JVM-et futtató gépen parancshívást java kódból:

http://www.rgagnon.com/javadetails/java-0014.html

Így kérdezhetjük le egy windows-os gép CPU kihasználtságát parancssorból:

http://www.sitaram-pamarthi.com/2009/09/how-to-get-cpu-utilization-of-remote.html

3.3 A metamodell megtervezése és példánymodell létrehozása

A modelltervezési feladat a hallgató feladata, az implementációs részleteket a „Szabályalapú üzleti logika" c. gyakorlat anyaga fedi le. A metamodell megtervezése és implementálása után gondoskodni kell a példánymodell létrehozásáról és a monitorozandó attribútumok beszúrásáról a példány modellbe.

Ez a gyakorlatban azt jelenti, hogy például 30 másodpercenként lekérdezzük a szerver memóriahasználatát és annak értékét beszúrjuk a szervert reprezentáló POJO-ba. Ezután update metódust kell végezni az objektumokon, hogy a Drools is értesüljön a változásokról. Ehhez az összes Drools által kezelt POJO és azok FactHandle-jai rendelkezésre kell álljanak, például egy HashMap-ben. Egy példa-kódrészlet, hogyan lehet ezt megvalósítani:

```
for (Map.Entry<ConfigurationItem, FactHandle> entry
            : map.entrySet()) {
      //key of a map element
      ConfigurationItem ci=entry.getKey();
      //value of a map element
      FactHandle fh=entry.getValue();

      //get the actual resource load from
      //the monitoring application
      double load=MonitoringDummy.getLoad();

      //set the load of the resource
      ci.setLoad(load);

      //inform the drools session
      //about the change
      ksession.update(fh, ci);
      //fire rules if possible

      //update chart (add 1 data to the series)
      mychart.updateChart(load);
}

ksession.fireAllRules();
```

Egy map-ben tároljuk az objektumokat és a hozzá tartozó FactHandle-t, egyesével végigmegyünk a map-on, lekérdezzük az aktuális terhelést, beállítjuk, majd értesítjük a Drools sessiont a változásról. Utána frissítjük a diagrammot is. Végül tüzeljük az érvényessé vált szabályokat, ha van ilyen.

3.4 Felügyeleti szabályok

A szabályok kitalálása és létrehozása szintén a hallgató feladata, szintén segít az implementációban a „Szabályalapú üzleti logika" c. gyakorlat anyaga. Elvárjuk kreatívabb szabályok kitalálását is, melyek alacsony és magas metrikákat egyaránt használnak. A szabályokhoz indoklást vagy magyarázatot várjunk.

3.5 Erőforráshasználat idősoros ábrázolása

A JFreeChart könyvtárt ajánljuk diagramok megjelenítésére. Erősen ajánlott a JFreeChart demók kipróbálása otthon, mérés előtt. A következő honlapon találhatunk egy demót, ami bemutatja, hogy hogyan kell JFreeChart segítségével egy egyszerű idősoros diagramit készíteni:

http://www.java2s.com/Code/Java/Chart/JFreeChartTimeSeriesDemo10withperminutedata.htm

Először lássuk a main függvényt. Ez lényegében csak létrehoz egy példányt, majd megjeleníti (az osztály az ApplicationFrame leszármazottja):

```
public static void main(final String[] args) {

    final TimeSeriesDemo demo = new TimeSeriesDemo10("Time Series Demo");
    demo.pack();
    RefineryUtilities.centerFrameOnScreen(demo);
    demo.setVisible(true);

}
```

A konstruktor függvény:

```
public TimeSeriesDemo10(final String title) {

    super(title);

    final TimeSeries series = new TimeSeries("Per Min Data", Minute.class);
    final Hour hour = new Hour();
    series.add(new Minute(1, hour), 10.2);
    series.add(new Minute(3, hour), 17.3);
    series.add(new Minute(9, hour), 14.6);
    series.add(new Minute(11, hour), 11.9);
    series.add(new Minute(15, hour), 13.5);
    series.add(new Minute(19, hour), 10.9);
    final TimeSeriesCollection dataset = new TimeSeriesCollection(series);
    final JFreeChart chart = ChartFactory.createTimeSeriesChart(
        "Time Series Demo 10",
        "Time",
        "Value",
        dataset,
        true,
        true,
        false
    );
    final ChartPanel chartPanel = new ChartPanel(chart);
    chartPanel.setPreferredSize(new java.awt.Dimension(500, 270));
    setContentPane(chartPanel);
}
```

Először egy idősort készítünk (TimeSeries), melybe egy idősor elemeit fogjuk egymás után belepakolni. Utána az idősorhoz hozzáadunk értékeket. Ezután egy adatsort hozunk létre (dataset). Az adatsor egy vagy több idősort képes tárolni. Ezután létrehozzuk a JFreeChartot, majd hozzáadjuk egy panelhez. Innen már magától értetődő a kód. Itt jegyeznénk meg, hogy ha az idősorhoz újabb elemet adunk hozzá, az automatikusan megjelenik a diagramon.

A következő hivatkozásban megtaláljuk kommentként, hogy a JFreeChart osztály egy példányát hogyan kell létrehozni, és mik a paraméterek, pár alapértelmezéstől eltérő megjelenítési kódrészletet is bemutat, valamint azt, hogyan lehet gombnyomással újabb adatsort hozzáadni egy meglévő diagramhoz.

http://sites.google.com/site/drjohnbmatthews/jfreechartdemo

4 Ellenőrző kérdések

A beugró kérdések kizárólag ebben a dokumentumban leírtakat fogják visszakérdezni. A mérés sikeres elvégzéséhez azonban szükség lesz a JMS mérés implementációja és a Drools mérés során elsajátított ismeretekre.

- Az üzleti folyamat mely részére (taskba) rakná a generált terhelés, és miért? Éles rendszerekben szükség van / lehet erre?
- Hogy nézne ki a műterhelés implementációja? Milyen erőforrás(oka)t fogyasztana?
- Milyen metrikákat célszerű mérni egy üzleti folyamatban? Hogyan mérné ezeket (implementáció-vázlat elég)?
- Soroljon fel magas (üzleti logikai szintű) és alacsony szintű (erőforrás közeli) metrikákat.
- Hogyan juttatná el a mért metrikákat a felügyeleti eszköznek, és mi lenne az célszerűen?
- Írja le egy üzleti folyamat felügyeletéhez kapcsolódó szabályt (pszeudo nyelven)!
- Írjon le egy üzleti folyamat felügyeletéhez kapcsolódó szabályt Drools nyelven, vázlatosan!
- Vázlatosan írja le, hogyan nézne ki egy vagy több paraméter idősoros megjelenítése JFreeChart használatával?
- Rajzoljon vázlatos architektúra ábrát, ami ábrázolja a mérendő üzleti folyamatot, a (mű) terhelés helyét, a mérendő paramétereket, azok felügyelő rendszerbe való átvitelét és a szabály-kiértékelő rendszert. Érdemes-e megjelenítést kapcsolni ehhez a rendszerhez, ha igen, hogyan?