EDA cache

EDA pre zrýchlenie výpotov vo vekej miere dokáže využíva rôzne typy cachovania. Cachujú sa definície databázových entít (vektory, scenáre, skupiny), dáta zadávaných vektorov, aj výsledky výpotu vypoítaných vektorov. Pri použití monolitickej EDA knižnice je možné používa klientske cache. Pri použití EDA klient/server je na serveri možné povoli globálnu cache a navyše využíva aj klientske cache.

Klientska cache

Klientska cache je viazaná na klienta, iže je použitená len klientom (procesom, ktorý používa EDA knižnicu), ktorý ju vytvoril.

Rozlišujú sa tri typy klientskej cache:

1. **0 - Read** – obsahuje len naítané dáta. Zápis dát obsah cache nemení.
2. **1 - Write_Back** – zapísané dáta sú odpamätané len v cache a do databázy sú uložené až na požiadanie.
3. **2 - Write_Through** – cache plnená pri ítaní aj zápise dát. Dáta sa zapisujú do cache aj do databázy.

Klientska cache vzniká pri volaní funkcie EDA_CreateCache, ktorá vracia identifikátor takto vytvorenej cache. Volaním funkcie EDA_EnableDefaultCache sa vytvorí predvolená *Read cache* s identifikátorom 0. Predvolená cache sa automaticky použije pre všetky operácie, ktoré nemajú explicitne zadaný identifikátor cache, ktorú majú použi.

V klientskej cache sú odpamätávané definície databázových entít, dáta zadávaných vektorov, aj výsledné dáta vypoítaných vektorov. Vzhadom na to, že výsledok vypoítaného vektora môže by závislý na parametroch, s ktorými bol ítaný (vrátane poiatoného a koncového asu ítaného intervalu), sa nacachovaný výsledok vypoítaného vektora použije len v prípade, že sa vektor druhýkrát naítava s rovnakými vstupnými parametrami.
V prípade *Write cache* sa výsledky vypoítaných vektorov (ktoré nie sú predpoítané) štandardne necachujú – je potrebné zavola EDA_EnableCacheV_V.
Vekos klientskej cache je obmedzená na maximum zadané pri jej vytváraní. Pri presiahnutí vekosti sú automaticky z cache odstránené najdlhšie nepoužité dáta.

Poas práce s *Write cache*, je možné opakovane vola EDA_FlushCache, o spôsobí zápis zmenených dát a uvonenie pamäte použitej na ich cachovanie. Ukonenie práce s cache je realizované volaním funkcie EDA_CloseCache.

Špeciálnym typom cache je **Bypass cache**, ktorá má identifikátor -1. Pri použití tejto cache sú pri práci s EDA obídené všetky klientske aj globálna cache a pracuje sa vždy s dátami z databázy a vypoítané vektory sa vždy prepoítavajú.

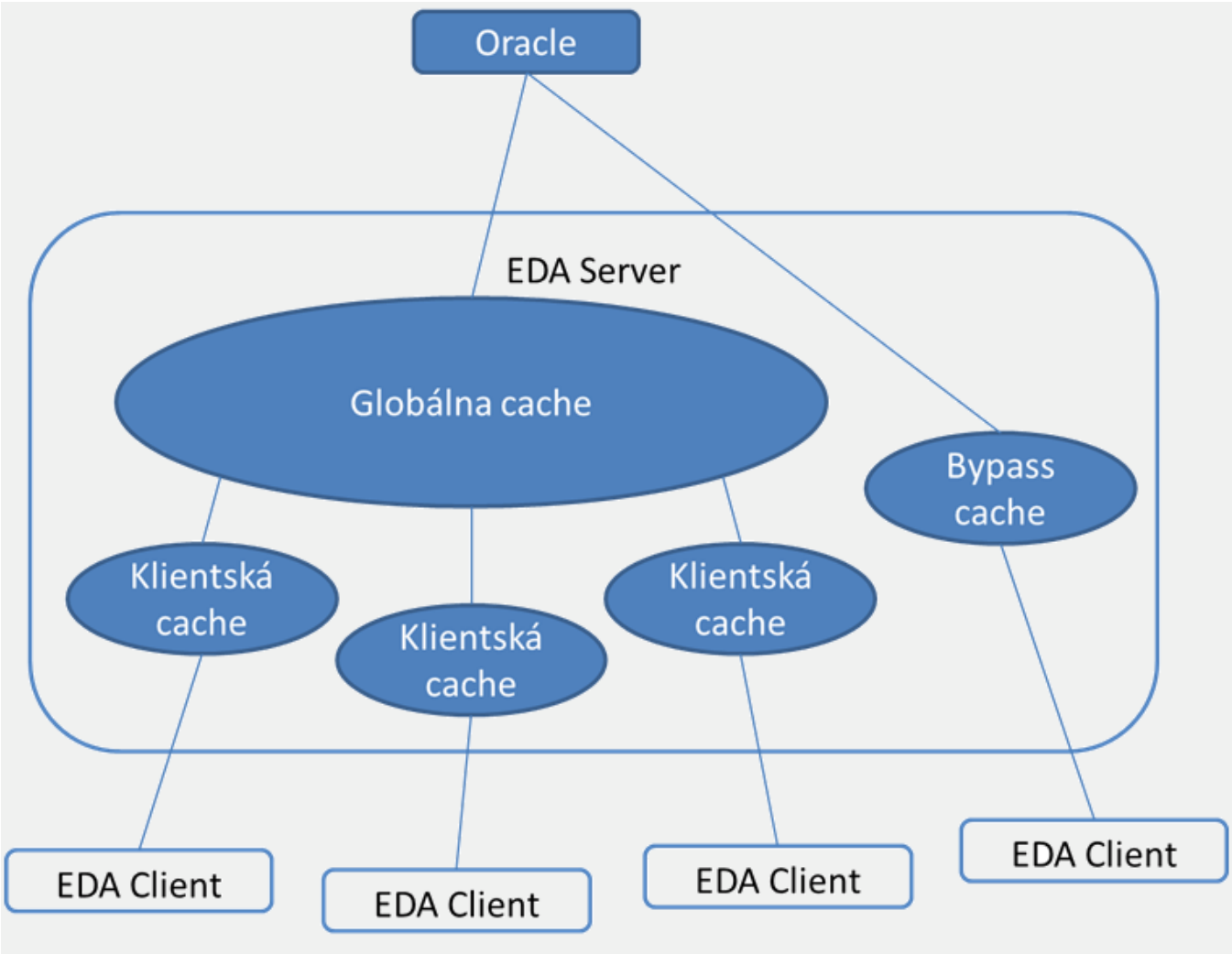
V prípade použitia EDA servera je cache pre klientov predalokovaná na dané maximum (štartovací parameter /EDACSC). Použitie predalokovaných blokov cache zvyšuje pamäovú stabilitu procesu a minimalizuje fragmentáciu operanej pamäte. Po vytvorení klientskej cache sa berú bloky cache z tohto predalokovaného množstva až do maxima definovaného pri vytvorení cache volaním EDA_CreateCache alebo do minutia predalokovaných blokov. V prípade, že už nie sú k dispozícii žiadne predalokované bloky, tak aj ak vekos klientskej cache nedosiahla svoje maximum, sú z cache vyhodené najdlhšie nepoužité bloky cache. Pri konfigurácii EDA servera je potrebné správne odhadnú vekos cache pre všetkých pripájaných klientov. Napr. ak je predpokladaný poet súasne pripojených klientov na EDA server 10 a každý z nich reálne využije 200MiB cache, musí ma EDA server na bezproblémovú obsluhu klientov predalokovaných 10 * 200 MiB = 2000MiB cache.

Príklad práce s cache:

```
BEGIN
 TEXT _vectorName = "vector.test"
 INT _vectorId = 1000000
 TIME _bt = %TimeFromItems(2015, 1, 1, 0, 0, 0, 0)
 TIME _et, _btDel, _etDel
 INT _errorCode, _i, _cacheId
 RECORD NOALIAS (SD.EDA_Arr_Obj) _data
 RECORD NOALIAS (SD.EDA_CreateVector_Params_V1) _createParams
 RECORD NOALIAS (SD.EDA_ReadValuesFromVektor_Params_V1) _readParams
 RECORD NOALIAS (SD.EDA_InsertValuesToVektor_Params_V1) _insParams
 RECORD NOALIAS (SD.EDA_DeleteIntervalFromVektor_Params_V1) _delParams
 RECORD NOALIAS (SD.EDA_CloseCache_Params_V1) _closeCacheParams

 ; vytvori minutovy periodicky vektor
 _createParams[1]^structVersion := 1
 _createParams[1]^periodBeginTime := _bt
 _createParams[1]^periodStepBase := 1
 _createParams[1]^periodStepCount := 60
 CALL %EDA_CreateVectorRec(_vectorName, _vectorId, _vectorName, 12, _createParams, _errorCode)
 IF _errorCode != 0 THEN
  RETURN
 ENDIF

 ; vytvori write-back cache
 CALL %EDA_CreateCache(1, 1000000, _cacheId, _errorCode)
 IF _errorCode != 0 THEN
  RETURN
 ENDIF

 ; vlozi data do cache
 REDIM _data[1440]
 FOR _i RANGE _data DO_LOOP
 _data[_i]^val := _i TIME (_bt + 60 * (_i - 1))
 END_LOOP
 _insParams[1]^structVersion := 1
 _insParams[1]^cacheId := _cacheId
 CALL %EDA_InsertValuesToVektorRec(_vectorName, _data, _insParams, _errorCode)
 IF _errorCode != 0 THEN
  RETURN
 ENDIF

 ; zmaze cast dat z cache
 _delParams[1]^structVersion := 1
 _delParams[1]^cacheId := _cacheId
 _btDel := %AddIntervalMono(_bt, 720 * 60)
 _etDel := %AddIntervalMono(_btDel, 3600)
 CALL %EDA_DeleteIntervalFromVektorRec(_vectorName, _btDel, _etDel, _delParams, _errorCode)
 IF _errorCode != 0 THEN
  RETURN
 ENDIF

 ; nacita data z cache
 _et := %AddIntervalMono(_bt, 86399)
 _readParams[1]^structVersion := 1
 _readParams[1]^cacheId := _cacheId
 REDIM _data[0]
 CALL %EDA_ReadValuesFromVektorRec(_vectorName, _bt, _et, 0, _readParams, _data, 1, _errorCode)
 IF _errorCode != 0 THEN
  RETURN
 ENDIF

 ; zatvori cache a zaroven zapise data do databazy
 _closeCacheParams[1]^structVersion := 1
 _closeCacheParams[1]^applyChanges := @TRUE
 CALL %EDA_CloseCacheRec(_cacheId, _closeCacheParams, _errorCode)
 IF _errorCode != 0 THEN
  RETURN
 ENDIF
END
```

Globálna cache

Globálna cache je volitená súas EDA servera a je zdieaná medzi všetkými klientmi pripojenými na EDA server. Vekos, resp. existencia i neexistencia globálnej cache je riadená štartovacími parametrami /EDACSG, z ktorej sa odvodí poet predalokovaných blokov globálnej cache. V globálnej cache sú automaticky cachované všetky definície databázových entít, s ktorými pracujú klienti EDA servera a všetky naítané dáta zadávaných vektorov.
Správa globálnej cache je plne automatická a pre klienta transparentná. Pri presiahnutí maximálnej vekosti cache sú z cache odstránené najdlhšie nepoužité dáta. Pri zmene dát, ktoré sa nachádzajú v globálnej cache, sú tieto dáta automaticky z cache odstránené. Na explicitné odstránenie dát z globálnej cache EDA servera je možné použi funkciu EDA_InvalidateGlobalCache.



Prednaítanie a vynútenie existencie vektora v globálnej cache je možné štartovacím parametrom EDA Servera /EDACIF, ktorým sa zadá názov inicializaného súboru pre globálnu cache. Cesta k súboru je relatívna vzhadom na aplikaný adresár D2000. Súbor na každom riadku obsahuje kód vektora, ktorý sa automaticky pri štarte EDA Servera naíta za celé obdobie platnosti do globálnej cache. Takto naítaný vektor nebude z globálnej cache odstránený automatickým mechanizmom, ktorý odstrauje najdlhšie nepoužité vektory. Vektor bude z cache odstránený len pri jeho zmene. Neexistujúce vektory alebo vektory, ktoré nemôžu by v globálnej cache sú ignorované.
Prednaítanie vektorov do cache ma význam pre vektory, ktoré sú modifikované len zriedkavo, ale ítané sú vemi asto a rýchle naítanie je potrebné už pri prvej aplikanej požiadavke. Typickým príkladom sú napríklad vektory kalendárov.

Vekos cache

Aby mala cache reálny prínos, musí by vhodne zvolená jej vekos. V prípade, že cache nie je dostatone veká na uloženie naítaných dát, dáta nebudú nacachované. V prípade, že Write cache nie je dostatone veká na vloženie daných dát, je generovaná chyba ERR_CACHE_NO_MEMORY. Vhodnú vekos cache je možné odhadnú na základe potu hodnôt, ktoré v nej majú by držané. Jedna hodnota vektora zaberá v cache približne 18 bajtov. Preto, ak je potrebné cachova milión hodnôt, bude pre dáta v cache potrebných viac ako 18MB cache. Do vekosti cache sú okrem samotných dát zapoítavané aj rôzne definície entít a pomocné informácie. Samotné dáta predstavujú zhruba 80% celkovej vekosti cache. Pre 18MB dát bude teda potrebná cache vekosti 18/0,8 = 22,5MB. Dáta vektorov sú navyše ukladané v blokoch po (štandardne) 512 hodnôt. V najhoršom prípade, ke je potrebné uloži jednu hodnotu z milióna vektorov alebo ukladané hodnoty netvoria súvislý interval, môžu nároky na cache narás až 512-krát.

Súvisiace stránky:

EDA cache funkcie