

# PostgreSQL Performans Ayarları ve İstatistikleri

Bu makale de PostgreSQL 8.3 için temel performans ayarlarından ve istatistik toplamadan bahsedeceğim.

[http://www.enderunix.org/docs/postgresql/postgresql\\_tuning.pdf](http://www.enderunix.org/docs/postgresql/postgresql_tuning.pdf) adresinden ulaşabilirsiniz.

Yazar: İsmail YENİGÜL

EnderUNIX Çekirdek Takım Üyesi

ismail at enderunix dot org

ismail nokta yenigul et endersys nokta com

<http://www.enderunix.org>

Değişiklikler

22 Nisan 2008 Pazartesi 14:32 - İlk Yazım

## İçindekiler

PostgreSQL Performans ayarları .....	2
1. Donanım Tercihi .....	2
1.1. Disk ve RAID .....	2
1.2. İşlemci .....	2
1.3. Shared Memory Parametreleri .....	2
1.4. Dosya Sistemi Parametreleri .....	3
2. PostgreSQL parametreleri .....	3
2.1. max_connection .....	3
2.2. shared_buffers .....	3
2.3. work_mem .....	3
2.4. effective_cache_size .....	4
2.5. Vacuum / autovacuum .....	4
3. Veritabanının İzlenmesi .....	5
3.1. İstatistiklerin Toplanması .....	5
3.2. pg_locks tablosu .....	6
4. Kaynaklar .....	7

# PostgreSQL Performans ayarları

Bu makalede PostgreSQL performansını artırmak için neler yapılacağına değineceğim. Testlerimde aşağıdaki donanıma sahip bir 1U sunucuyu kullandım.

İşlemci: Intel(R) Core(TM)2 Duo CPU E4500  
Bellek: 2GB DDR2 667 Mhz  
Disk: 2 x 80 GB SATA II (Onboard RAID 1)  
İşletim Sistemi: FreeBSD 6.3 – RELEASE amd64  
PostgreSQL versiyonu: 8.3

## 1. Donanım Tercihi

### 1.1. Disk ve RAID

Veritabanı için en uygun RAID tipi RAID 10 veya RAID 1'dir. RAID 5 veritabanları için çok yavaş kaldığından kullanılması tavsiye edilmemektedir. Mümkünse WAL dizinini RAID 1, data dizinini RAID 10 üzerinde olmalıdır. Disk olarakda mümkünse "battery backend" özelliği olan diskler alınmalıdır.

### 1.2. İşlemci

İşlemci sayısı  $\leq 8$  ve tip olarak Xeon,Opteron işlemcilerin kullanılması tavsiye edilir.

## 2. İşletim Sistemi Parametreleri

### 2.1. Shared Memory Parametreleri

PostgreSQL performansında shared memory değerlerin oldukça önemli etkisi vardır. Bu yüzden shared memory değerleri ne kadar çok ise performansta buna oranla artacaktır. Genellikle sadece veritabanı olarak çalışan sistemlerde belleğin yarısı kadar shared memory değeri tanımlanabilir. FreeBSD'de 1GB shared memory ayarı yapmak için /etc/sysctl.conf dosyasına aşağıdaki değerler eklenir.

```
#1024 MB shmmax  
kern.ipc.shmmax=1073741824
```

```
#shmalls = shmmax / 4096 (page size)  
kern.ipc.shmall=262144  
kern.ipc.semmsl=512  
kern.ipc.semmap=256
```

Ayrıca /boot/loader.conf dosyasına aşağıdaki satırlar eklenir.

```
kern.ipc.semmsl=512  
kern.ipc.semms=1024  
kern.ipc.semmsu=512  
kern.maxusers=512
```

/boot/loader.conf dosyasındaki deęişikliklerin etkinleşmesi için sistemin reboot edilmesi gerekir.

## 2.2. Dosya Sistemi Parametreleri

PostgreSQL verileri /db isimli ayrı bir disk bölümü altına atıldı. Bu bölüm noatime ve atime ile ayrı ayrı bağlandı. Yapılan test sonuçlarına göre disk bölümünü noatime olarak bağlamak(mount), %2 ile %5 arası bir performans artışı sağlıyor.

## 3. PostgreSQL parametreleri

Postgresql.conf dosyasında aşağıdaki parametreler PostgreSQL performansında önemli rol oynamaktadır.

### 3.1. *max\_connection*

Aynı anda kaç bağlantı kabul edileceğini belirler. Yukarıdaki sysctl.conf ve /boot/loader.conf dosyasındaki deęerlerle max\_connection sayısı en fazla 950 olabiliyor. Siz bu deęeri ihtiyacınıza göre 100 ile 950 arasında bir deęer tanımlayabilirsiniz.

Not: Her bir bağlantı 400 bytes civarında bir shared memory'ye ihtiyaç duymaktadır.

### 3.2. *shared\_buffers*

PostgreSQL'in kullanımına ayrılan shared memory miktarını belirler. Bu deęerin hesaplaması biraz karmaşık olduğundan genelde belleğin 1/4'ü ile 1/2'si arasında denemeler yaparak sisteminiz için en uygun deęeri bulabilirsiniz. Yukarıdaki belirtilen sunucu için bu deęer 500MB'den başlayıp 1GB'ta kadar çıkabilir.

### 3.3. *work\_mem*

PostgreSQL'in dahili sıralama işlemleri ve hash tablolarının kullandığı bellek miktarıdır. Ön tanımlı olarak 1MB'dir. Bu parametrenin deęeri 2MB olarak tanımlanabilir. Ayrıca bu parametre bağlantı sırasında da aşağıdaki gibi tanımlanabilmektedir.

```
db=# set work_mem = '5MB';  
SET
```

### 3.4. *effective\_cache\_size*

Postgres buffer'ı ile işletim sistemi buffer'ı arasındaki efektif cache miktarını belirleyen parametredir. Genellikle belleğin yarısı kadar bir deęer verilir. Bu deęer postgresql'in belleği nasıl kullanacağını deęiştirmez. Postgres'in tarama tipini (indeks

tarama, sıralı tarama) seçmesinde yardımcı olur.  
effective\_cache\_size = 1024MB

### **3.5. Vacuum / autovacuum**

Vacuum, PostgreSQL veritabanı bakımı araçlarından biridir. Vacuum işleminin düzenli aralıklarla yapılması tavsiye edilmektedir. Vacuum sayesinde aşağıdaki faydalar elde edilir.

#### **1. Silinen ve güncellenen satırlar için ayrılan disk alanının boşaltılması:**

Normalde PostgreSQL, UPDATE ve DELETE işlemlerinde satırın eski sürümü hemen silmez. Bu işlem MVCC(Multiversion Concurrency Control) özelliğinden faydalanmak için gerekmektedir. Satırın eski sürümü diğer transkiyonlarda gözükme ihtimalinden dolayı silinmemelidir. Fakat bu, o versiyonun sonsuza kadar kalacağı manasına gelmemektedir. Düzenli olarak (örneğin, her gece) çalıştırılacak bir vacuum işlemi ile kullanılmayan eski satır versiyonlarının silinmesi sağlanır. Buradanda anlaşılacağı üzere, çok sık değişen tablolar üzerinde daha sık vacuum işleminin yapılması gerekir.

#### **2. Planlayıcı İstatistiklerinin Güncellenmesi:**

PostgreSQL'in sorgu planlayıcısı(query planner), tablonun içerisindeki bilgilerin istatistiklerine göre en uygun planlayıcısı seçmektedir. Bu istatistikler ANALYZE komutu ile elde edilmektedir. Bu komut tek başına veya VACUUM komutuna parametre olarak çalıştırılabilmektedir. Eğer istatistikler düzenli olarak güncellenmese, PostgreSQL planlayıcı seçiminde sağlıklı karar veremeyebilir. Bu da performansı olumsuz yönde etkileyecektir.

#### **3. Transaction ID değerinin başa dönmesini(wraparound) engellemek:**

PostgreSQL MVCC transaksiyon semanteği, transaction ID (XID) sayılarının karşılaştırmasına dayanmaktadır. Yeni eklenen bir satırın XID değeri, şu anki transaksiyonun XID değerinden büyükse satırın bu versiyonu şu anki transaksiyonda gözükmemelidir. XID değeride 32 bit olduğundan dolayı bir süre sonra XID değeri sıfırlanıp başa dönebilir. Bu durumda yeni XID değeri eskisinden küçük gözükeceğinden, yeni sorgularda veriler veritabanında olsa bile çıktıda gözükmeyecektir. Bunu engellemek için düzenli olarak vacuum işlemi yapılmalıdır. Konuyla ilgili detaylar için PostgreSQL dökümanlarına bakabilirsiniz.

PostgreSQL 8.1'den itibaren gelen ayrı bir autovacuum servisi(daemon) ile vacuum işlemlerinin ihtiyaç duyulduğu anda düzenli olarak yapılmasını sağlar. Autovacuum özelliği devreye alındığında, servis düzenli olarak tabloları kontrol edecektir. Bu özelliği kullanabilmek için autovacuum'un ihtiyaç duyduğu satır seviyesindeki istatistiklerin devreye alınması gerekir. (stats\_start\_collector ve stats\_row\_level) değerlerinin true yapılması gerekir.

## 4. Veritabanının İzlenmesi

### 4.1. İstatistiklerin Toplanması

PostgreSQL'de istatistikleri toplamak için gömülü olarak birkaç özellik gelmektedir. Bu özellikleri devreye almak için postgresql.conf dosyasında stats\_start\_collector parametresinin değeri true olmalıdır. Buna ek olarak stats\_block\_level ve stats\_row\_level parametreleri ne kadar detaylı istatistikler toplanacağını belirler. Son olarak da stats\_command\_string değeri true yapılırsa o anda çalıştırılan sorgu da istatistiklerde gözükecektir.

İstatistikler aşağıdaki görüntü(view) üzerinden izlenebilir.

1. **pg\_stat\_activity**: Her bir satırda bir prosese ait veritabanı OID değeri, veritabanı adı, süreç numarası vs. gibi bilgileri gösterir.

```
ecm=# select * from pg_stat_activity;
```

datid	datname	procpid	usesysid	username	current_query	waiting	query_start	backend_start	client_addr	client_port
16384	ecm	10091	10	nxpgsql	<IDLE>	f	2008-04-21 19:28:22.25577+03	2008-04-21 19:28:22.252707+03	127.0.0.1	59571
16384	ecm	10301	18038	iso	select * from pg_stat_activity;	f	2008-04-21 19:46:16.255084+03	2008-04-21 19:32:06.047707+03		-1
....										

2. **pg\_stat\_database**: Her bir veritabanına ait bağlantı bilgilerini gösterir:

```
ecm=# select * from pg_stat_database;
```

datid	datname	numbackends	xact_commit	xact_rollback	blks_read	blks_hit
10819	postgres	0	0	0	0	0
16384	ecm	11	0	0	0	0
1	template1	0	0	0	0	0
10818	template0	0	0	0	0	0

(4 rows)  
ecm=#

3. **pg\_stat\_all\_tables**: Her bir tablo hakkında istatistik bilgileri sunar.

```
ecm=# select * from pg_stat_all_tables;
```

```

    relid |      schemaname       |      relname       | seq_scan | seq_tup_read |
    idx_scan | idx_tup_fetch | n_tup_ins | n_tu
p_upd | n_tup_del | last_vacuum | last_autovacuum | last_analyze | last_autoanalyze
-----+-----+-----+-----+-----+-----
-----+-----+-----+-----+-----+-----
    17479 | public              | vcat_defs          |          |          |
    0 |          0 |          0 |          |          |
          |          |          |          |          |
    16649 | pg_toast            | pg_toast_16647    |          |          |
    0 |          0 |          0 |          |          |
          |          |          |          |          |
    17271 | public              | policy_element     |          |          |
    0 |          0 |          0 |          |          |
          |          |          |          |          |
    0

```

Buna benzer daha bir çok nesne bulunamaktadır. Tüm listesi PostgreSQL dökümanlarında bulunabilir.

## 4.2. *pg\_locks* tablosu

Bu tabloda sorgu yapılarak o anda bekleyen kilitleri(locks) görebilir. Örnek:

```

ecm=# select * from pg_locks;

   locktype      | database | relation | page | tuple | transactionid | classid | objid
| objsubid | transaction | pid | mode | granted
-----+-----+-----+-----+-----+-----+-----+-----
relation          |          | 16384 | 10328 |        |          |          |
|          | 264561 | 10301 | AccessShareLock | t
transactionid    |          |          |        |        |          |          |
|          | 264561 | 10301 | ExclusiveLock | t          | 264561 |

(2 rows)

ecm#

```

## 5. Kaynaklar

Practical PostgreSQL kitabı - <http://www.commandprompt.com/ppbook/book1>  
 PostgreSQL El kitabı: <http://www.postgresql.org/docs/8.3/interactive/index.html>  
 RAID 10: [http://www.acnc.com/04\\_01\\_10.html](http://www.acnc.com/04_01_10.html)  
<http://archives.postgresql.org/pgsql-performance/2005-06/msg00477.php>  
<http://www.westnet.com/~gsmith/content/postgresql/pg-5minute.htm>