Linux and Oracle TPC-C Optimizations When Implementing the Seagate® Nytro™ Flash Accelerator Card

Technology Paper

Author:
Rick Stehno
Principal Database Engineer
Seagate

Introduction

The Nytro flash accelerator card is most often purchased to increase database performance, such as the enterprise Oracle database which runs on Linux. Once the Nytro card has been installed, the most logical next question would be, "How do I set up the infrastructure in order to get the best performance?" As each organization and environment is unique, no cookie cutter questions will reveal specifically how the Nytro card should be configured for optimum performance, but there are tools available and best practices to assist with this effort.

Most of the popular relational and nonrelational databases run on many different operating systems. This guide will describe Linux-specific configurations, but most other operating systems should also work with Nytro cards.

This document will present tuning and processes that can be applied to increase TPC-C performance when using the Nytro accelerator card. The performance optimization recommendations that this document will review includes the following:

- Nytro flash accelerator card configuration
- Linux configuration
- Oracle database configuration

Many of these optimizations were centered on increasing concurrency, decreasing locks, and allowing more physical I/O to the Nytro card when used in an Oracle 11gR2 or later database.

Optimizing Performance with the Nytro Flash Accelerator Card

Aligning the Nytro Flash Accelerator Card

The most important step to perform on the Nytro card is to create a partition that is aligned on a specific boundary (such as 4k or 8k) so each read and write to the flash device will require only one physical input/output (I/O) operation. If the Nytro card is not partitioned on such a boundary, then reads and writes will span the sector groups which doubles the I/O latency for each read or write request.

To create a partition, use the `sfdisk` command to start a partition on a 1M boundary (sector 2048). Aligning to a 1M boundary resolves the dependency to align to a 4k, 8k, 64k and other boundaries divisible by 4.

Prior to this step, several questions should be posed about your specific deployment and how you are going to use this device. Will this be used with Oracle's Automatic Storage Management (ASM), a standalone filesystem partition, part of a logical volume or part of a RAID group?

Identifying the Appropriate Nytro Flash Accelerator Card

Typically when deploying the Nytro card for database caching (for example, the Database Smart Flash Cache feature in the Oracle database), a single-partitioned Nytro card would be suitable if the capacity is sufficient to meet the needs of the database now and over the next several years. In this case, the `sfdisk` command to create the partition would be:

```
echo "2048,," | sfdisk -uS /dev/sdX -force
```

NOTE: If using ASM, skip to the *Using ASM* section.

If using a filesystem instead of ASM, and the need exists to deploy multiple Nytro cards for database caching, create a Logical Volume Manager (LVM) over multiple Nytro cards to simplify administration. The `sfdisk` command to create a partition for each Nytro card is:

```
echo "2048,,8e" | sfdisk -uS /dev/sdX --force
```

"`8e`" is the system partition type for creating a logical volume.

None of the above solutions requires fault tolerance since they will be used for write-through caching, meaning data will be transparent between disk and cache.

If the Nytro card will be used for persisting data, then multiple Nytro cards will be needed to provide fault tolerance. By using two or more Nytro cards to build the RAID array or ASM disk group, the following concepts will eliminate any single point of failure. There are a number of ways to create a redundant ASM disk group or RAID over multiple Nytro cards, some examples are:

- Use LVM with the RAID option.
- Use the software RAID utility MDADM to create the RAID array.
- ASM Normal/High Redundancy option

Identifying the Best RAID Level

Oracle coined the term SAME, which means Stripe And Mirror Everything, in 1999 and popularized the practice which many database administrators (DBA) and storage administrators continue to follow.

To implement SAME, first determine how the Nytro cards will be accessed. This could include:

- Small random reads and writes
- Larger sequential reads
- Hybrid (mix of both)

In database deployments, your choice is usually among online transaction processing (OLTP) applications such as airline and hotel reservation systems, corporate financial or enterprise resource planning (ERP) applications or analytical/data warehouse/data mining applications (DW), or a mix of these environments. OLTP applications involve small random reads and writes as well as many sequential writes for log files. Data warehouse/analytical/data mining applications involve mostly large sequential reads with very few sequential log writes.

Before setting up one or many Nytro cards in a RAID array, either using LVM on RAID or creating a RAID array using multiple device administration (MDADM), it is important to understand the access pattern of the I/O, capacity requirements and budget. These requirements will dictate which RAID level will work best for the specific environment.

RAID options would include either a RAID 1/RAID 10 configuration (mirroring without striping, or striping and mirroring respectively), or RAID 5 (striping with parity). RAID 1/RAID 10 is a larger investment, but delivers the best performance, whereas RAID 5 costs less but imposes a significant write penalty. To optimize performance of an OLTP application, it is recommended to either implement a RAID 1 or RAID 10 array unless budget was a constraint, then RAID 5 should be considered. In a data warehouse/analytics environment with the majority of the I/O is reads, RAID 5 would be the option to implement. Knowing how to tune the configuration to the application is key to reaping the best performance.

For either RAID array, create an aligned partition using `sfdisk`:

```
echo "2048,,fd" | sfdisk -uS /dev/sdX --force
```

"`fd`" is the system identifier for a Linux RAID auto device.

Keep in mind that it is not mandatory to create a partition for LVMs or RAID arrays. Instead, RAW devices can be assigned. It's important to remember to align the sectors if combining RAW and partitioned devices or just creating a basic partition. It's sound practice to always create an aligned partition when using Nytro cards.

Aligned partitions have now been created and are ready to be used in LVMs or RAID arrays. Instructions for creating these are on the Web or in Linux/UNIX reference manuals. Below are several links that review the process of creating LVM, RAID or LVM on RAID:

https://raid.wiki.kernel.org/index.php/Partitioning_RAID_/_LVM_on_RAID

http://www.gagme.com/greg/linux/raid-lvm.php

Also, remember that when creating LVMs with striping or RAID arrays, it is important to specify a stripe width value. Oracle and EMC conducted several studies on this and concluded that a 1M stripe width performed the best as long as the database I/O request was equal to or less than 1M.

Filesystem Tuning

Deploying an XFS filesystem with a 4KB block size, as shown below, demonstrated improvement in overall performance. The EXT filesystem is limited to a single mutex per inode while XFS allows more advanced locking mechanisms, which is important when flushing data from the Oracle data buffer.

```
mkfs.xfs -s size=4096
```

When considering mount options, several options exist that can be applied to increase performance of the Nytro card. For both EXT-4 and XFS filesystems, the recommendations are:

For EXT-4: `noatime,nodiratime,max_batch_time=0,nobarrier,discard`

For XFS: `nobarrier,discard`

NOTE: The mount option `discard` could have negative or positive effects on the performance of your system. An alternative to setting the `discard` option is creating a batch job running the `fstrim` command to discard unused blocks in the system, so performance is only affected when this batch job is run which would normally be in a maintenance window. Other enterprise environments may not have such a window to run a batch job. Customers would benefit by implementing the mount `discard` option.

Using ASM

The following steps cover configuring a DISK for ASM. In order to use ASM, you will need to install the Oracle Grid software from otn.oracle.com with a recommendation of using Oracle ASMLIB when configuring ASM. Included in the box of the latest version of Oracle Linux, ASMLIB offers an easier way to configure ASM. If you are using an older version of ASM, you will need to install the RPMs for ASM from support.oracle.com.

1. Create aligned partition. Oracle recommends using the full disk for ASM, so just create one large aligned partition. This command is recommended:

   ```
   echo "2048,," | sfdisk -uS /dev/sdX -force
   ```

   If using one (1) or many Nytro cards with ASM, this command should be executed for each.

2. Create an ASM disk. Once the device has an aligned partition created on it, assign it to ASM by using the ASM `createdisk` command with two input parameters – ASM disk name and the PCIe flash partitioned device name – as follows:

   ```
   /etc/init.d/oracleasm createdisk ASMDISK1 /dev/sdX1
   ```

   If using one (1) or many Nytro cards with ASM, this command should be executed for each.

   To verify that the create ASM disk process was successful, and the device was marked as an ASM disk, enter the following commands:

   ```
   /etc/init.d/oracleasm querydisk /dev/sdX1
   ```

   The output should state: `/dev/sdX is an Oracle ASM disk [OK]`

   ```
   /etc/init.d/oracleasm listdisks
   ```

   The output should state: `ASMDISK1`

3. Assign ASM disk(s) to a disk group. The ASM disk group is the primary component of ASM as well as the highest level data structure in ASM. A disk group is a container of multiple ASM disks, and it is the disk group that the database references when creating Oracle Tablespaces or for the location for the Oracle Database Smart FlashCache.

   There are multiple ways to create an ASM disk group. The easiest way is to use ASM Configuration Assistant (ASMCA), which walks you through the creation process. See Oracle ASM documentation on how to use ASMCA.

   Here are the steps for creating a disk group:

   Log in to GRID using: `sqlplus / as sysasm.`

   Select name, path, header status and state from v$asm_disk as follows:

   ```
   NAME              PATH                      HEADER_STATU        STATE
   ---------------   -----------------------   -------------       ----------
   ASMDISK1          /dev/oracleasm/disks/D1   CANDIDATE           NORMAL

   create diskgroup DG1 external redundancy disk '/dev/oracleasm/disks/D1';
   ```

   If creating a disk group over multiple Nytro cards to meet capacity needs for the Database Smart Flash Cache, enter:

   ```
   create diskgroup DG1 external redundancy disk '/dev/oracleasm/disks/D1', '/dev/oracleasm/
   disks/D2';
   ```

   If creating a disk group for database data or indexes that needs to be fault tolerant, enter:

   ```
   create diskgroup DG1 normal redundancy
        failgroup fg1 disk '/dev/ oracleasm/disks/D1'
        failgroup fg2 disk '/dev/oracleasm/disks/D2';
   ```

The disk group is now ready to be used in creating an Oracle database tablespace or for using it as Database Smart Flash Cache. To use this disk group in an Oracle database, please refer to Oracle's database documentation at docs.oracle.com.

Tuning Linux

Many Linux variables exist that can be tuned to extract the best performance from the Nytro card. Some of these might perform better than others but when used as a whole, they will benefit in more mixed environments. These variables can be set in a number of different ways, but the recommendation is to use the script that is referenced in the next section on how to persist these variables across system reboots.

For transaction-based applications/databases, the following configuration is recommended:

```
echo "deadline" > /sys/block/sdX/queue/scheduler
echo 1 > /sys/block/sdX/queue/iosched/fifo_batch
echo 0 > /sys/block/sdX/queue/iosched/front_merges
echo 5 > /sys/block/sdX/queue/iosched/writes_starved
echo 2048 > /sys/block/sdX/queue/nr_requests
echo 1024 > /sys/block/sdX/queue/max_sectors_kb
echo 1024 > /sys/block/sdX/device/queue_depth
echo 0 > /sys/block/sdX/queue/nomerges
echo 0 > /sys/block/sdX/queue/rotational
blockdev --setra 0 /dev/sdX
echo 0 > /sys/block/sdX/queue/add_random
echo 2 > /sys/block/sdX/queue/rq_affinity
echo 1 > /sys/block/$variable1/queue/iosched/fifo_batch
echo 0 > /sys/block/$variable1/queue/iosched/front_merges
echo 5 > /sys/block/$variable1/queue/iosched/writes_starved
```

For data warehouse or data analytics types of applications/databases, the following recommendations are best suited for these environments:

```
echo "deadline" > /sys/block/sdX/queue/scheduler
echo 1 > /sys/block/sdX/queue/iosched/fifo_batch
echo 0 > /sys/block/sdX/queue/iosched/front_merges
echo 5 > /sys/block/sdX/queue/iosched/writes_starved
echo 2048 > /sys/block/sdX/queue/nr_requests
echo 1024 > /sys/block/sdX/queue/max_sectors_kb
echo 1024 > /sys/block/sdX/device/queue_depth
echo 0 > /sys/block/sdX/queue/nomerges
echo 0 > /sys/block/sdX/queue/rotational
blockdev --setra 4096 /dev/sdX
echo 0 > /sys/block/sdX/queue/add_random
echo 2 > /sys/block/sdX/queue/rq_affinity
echo 0 > /sys/block/$variable1/queue/iosched/front_merges
echo 10 > /sys/block/$variable1/queue/iosched/writes_starved
```

Invoke HugePages

Instead of Linux using 4k memory pages, Linux and Oracle can be configured to use HugePages which are 2M in size. Using HugePages will decrease the number of memory pages from 500 to 1 allowing Linux to operate more efficiently.

To check if the system is setup for HugePages, execute:

```
"cat /proc/meminfo | grep Huge*".
```

If any of the values are greater than 0, then HugePages has been modified to enable HugePages. Now we just have to see if the number of HugePages is large enough for the Oracle database.

NOTE: Oracle option AMM cannot be used in conjunction with HugePages. Oracle ASMM can be used in conjunction with HugePages.

To set up HugePages for Oracle, you need to calculate how much memory the Oracle database is using, which includes all its buffers and memory pools. To calculate the memory allocation that Oracle is taking up, create a script that can be executed:

```
hugepages_settings.sh
#!/bin/bash
#
# hugepages_settings.sh
#
# Linux bash script to compute values for the
# recommended HugePages/HugeTLB configuration
#
# NOTE: This script does calculation for all shared memory
# segments available when the script is run, no matter it
# is an Oracle RDBMS shared memory segment or not.
#
# This script is provided by Doc ID 401749.1 from My Oracle Support
# http://support.oracle.com
# Welcome text
echo "
```

This script is provided by Doc ID 401749.1 from My Oracle Support (http://support.oracle.com) where it is intended to compute values for the recommended HugePages/HugeTLB configuration for the current shared memory segments. Before proceeding with the execution please note following:

- For ASM instance, it needs to configure ASMM instead of AMM.
- The `'pga_aggregate_target'` is outside the SGA and you should accommodate this while calculating SGA size.
- In case you change the DB SGA size, as the new SGA will not fit in the previous HugePages configuration, it had better disable the whole HugePages, start the DB with new SGA size and run the script again.

And make sure that:

- Oracle Database instance(s) are up and running.
- Oracle Database 11g Automatic Memory Management (AMM) is not setup (See Doc ID 749851.1)
- The shared memory segments can be listed by command:

  ```
  # ipcs -m
  ```

Press Enter to proceed...”

```
read
# Check for the kernel version
KERN=`uname -r | awk -F. '{ printf("%d.%d\n",$1,$2); }'`
# Find out the HugePage size
HPG_SZ=`grep Hugepagesize /proc/meminfo | awk '{print $2}'`
if [ -z "$HPG_SZ" ];then
   echo "The hugepages may not be supported in the system where the script is being
   executed."
   exit 1
fi
# Initialize the counter
NUM_PG=0
# Cumulative number of pages required to handle the running shared memory segments for SEG_
BYTES in `ipcs -m | cut -c44-300 | awk '{print $1}' | grep "[0-9][0-9]*"`
do
   MIN_PG=`echo "$SEG_BYTES/($HPG_SZ*1024)" | bc -q`
   if [ $MIN_PG -gt 0 ]; then
      NUM_PG=`echo "$NUM_PG+$MIN_PG+1" | bc -q`
   fi
done
RES_BYTES=`echo "$NUM_PG * $HPG_SZ * 1024" | bc -q`
# An SGA less than 100MB does not make sense
# Bail out if that is the case
if [ $RES_BYTES -lt 100000000 ]; then
   echo "***********"
   echo "** ERROR **"
   echo "***********"
   echo "Sorry! There are not enough total of shared memory segments allocated for
HugePages configuration. HugePages can only be used for shared memory segments
that you can list by command:
   # ipcs -m
```

of a size that can match an Oracle Database SGA. Please make sure that:

- Oracle Database instance is up and running.
- Oracle Database 11g Automatic Memory Management (AMM) is not configured”

```
    exit 1
fi
# Finish with results
case $KERN in
  '2.4') HUGETLB_POOL=`echo "$NUM_PG*$HPG_SZ/1024" | bc -q`;
     echo "Recommended setting: vm.hugetlb_pool = $HUGETLB_POOL" ;;
  '2.6') echo "Recommended setting: vm.nr_hugepages = $NUM_PG" ;;
  *) echo "Unrecognized kernel version $KERN. Exiting." ;;
esac
# End
```

The output from the script above was:

Recommended setting: `vm.nr_hugepages = 44038`

To allow Oracle to lock memory, you need to grant it the privilege. Modify `/etc/security/limits.conf` and set the `memlock` parameter to 5GB, which is high for the 95GB SBA, but setting the value a little high doesn't hurt. The setting is in kilobytes.

```
oracle    soft   memlock      97280000000
oracle    hard   memlock      97280000000
```

Log out of Oracle, then log back on.

Shut down Oracle then restart it.

Look at alert log file to make sure hugepages is enabled:

```
****************** Large Pages Information *****************
Parameter use_large_pages = AUTO
Total Shared Global Region in Large Pages = 86 GB (100%)
Large Pages used by this instance: 44033 (86 GB)
Large Pages unused system wide = 0 (0 KB) (alloc incr 256 MB)
Large Pages configured system wide = 44033 (86 GB)
Large Page size = 2048 KB
Time taken to allocate Large Pages = 0.306200 sec
```

Persist the Linux Environment Variables for PCIe-based Devices after Reboots

NOTE: This section is used when using Linux filesystems and not ASM.

In a Linux server, there are times when device assignments change after reboots. Sometimes the Nytro card can be assigned `/dev/sda`. Other times it can be assigned `/dev/sdd` or any device name. This variability can wreak havoc when modifying the Linux environment variables. To avoid this issue, assignments using the SCSI address should be used so all of the Linux performance variables will be persisted properly across reboots.

NOTE: If using a filesystem, use the device UUID address in the mount statement in `/etc/fstab` so the `mount` command will be persisted across reboots.

The first step to solving this issue for assigning operating system assignments to the Nytro card is to use the following script, which can be copied and pasted into `/etc/rc.local` with the exception of SCSI address of the Nytro card that is required before executing the script.

The SCSI address below will need to be modified with the address of the Nytro card. To get this value, issue the following command:

```
ls -al /dev/disk/by-id
```

When the Nytro card is installed, Linux will assign a name to the device. For example, the device name can be listed as `/dev/sdX` and "`X`" can be any letter. The output from the '`ls`' command above will show the SCSI address for this Nytro card. Don't use the address that has '`-partX`' in it. Be sure to note this SCSI address as it will be required to create the script below.

NOTE: Include one space after the SCSI address before the quote.

Copy the code below and create a file called "nwd_getdevice.sh" with the modification of the SCSI address (as above).

`nwd_getdevice.sh`

```
ls -al /dev/disk/by-id |grep 'scsi-3600508e07e726177965e06849461a804 ' |grep /sd > nwddevice.
txt
awk '{split($11,arr,"/"); print arr[3]}' Nytro carddevice.txt > nwd1device.txt
variable1=$(cat nwd1device.txt)
echo "4096" > /sys/block/$variable1/queue/nr_requests
echo "512" > /sys/block/$variable1/device/queue_depth
echo "deadline" > /sys/block/$variable1/queue/scheduler
echo 1 > /sys/block/$variable1/queue/iosched/fifo_batch
echo 0 > /sys/block/$variable1/queue/iosched/front_merges
echo 5 > /sys/block/$variable1/queue/iosched/writes_starved
echo "2" > /sys/block/$variable1/queue/rq_affinity
echo 0 > /sys/block/$variable1/queue/rotational
echo 0 > /sys/block/$variable1/queue/add_random
echo 1024 > /sys/block/$variable1/queue/max_sectors_kb
echo 0 > /sys/block/$variable1/queue/nomerges
blockdev --setra 0 /dev/$variable1
```

After saving this file, change permission of the file to "execute" and then place this command in the `/etc/rc.local` file:

```
/path/nwd_getdevice.sh
```

To test this script, execute it on the command line exactly how it is stated it in the `rc.local` file. The next time the system is rebooted, the settings will be set to the appropriate device.

If you plan to deploy multiple Nytro cards in the server, the easiest way is to duplicate all of commands in the `nwd_getdevice.sh` script and append them at the end. Then change the SCSI address of the next card and overlay the SCSI address in the newly pasted area. You can follow this procedure for howeer many Nytro cards are installed in the server. An example of this could be:

`nwd_getdevice.sh`

```
ls -al /dev/disk/by-id |grep 'scsi-3600508e07e726177965e06849461a804 ' |grep /sd > nwddevice.
txt
awk '{split($11,arr,"/"); print arr[3]}' nwddevice.txt > nwd1device.txt
variable1=$(cat nwd1device.txt)
echo "4096" > /sys/block/$variable1/queue/nr_requests
echo "512" > /sys/block/$variable1/device/queue_depth
```

```
echo "deadline" > /sys/block/$variable1/queue/scheduler
echo 1 > /sys/block/$variable1/queue/iosched/fifo_batch
echo 0 > /sys/block/$variable1/queue/iosched/front_merges
echo 5 > /sys/block/$variable1/queue/iosched/writes_starved
echo "2" > /sys/block/$variable1/queue/rq_affinity
echo 0 > /sys/block/$variable1/queue/rotational
echo 0 > /sys/block/$variable1/queue/add_random
echo 1024 > /sys/block/$variable1/queue/max_sectors_kb
echo 0 > /sys/block/$variable1/queue/nomerges
blockdev --setra 0 /dev/$variable1
# *********
# second Nytro card
# *********
ls -al /dev/disk/by-id |grep 'scsi-2ndscsiaddr12345666666544444444444 ' |grep /sd > Nytro
carddevice.txt
awk '{split($11,arr,"/"); print arr[3]}' Nytro carddevice.txt > Nytro card1device.txt
variable1=$(cat nwd1device.txt)

echo "4096" > /sys/block/$variable1/queue/nr_requests
echo "512" > /sys/block/$variable1/device/queue_depth
echo "deadline" > /sys/block/$variable1/queue/scheduler
echo 1 > /sys/block/$variable1/queue/iosched/fifo_batch
echo 0 > /sys/block/$variable1/queue/iosched/front_merges
echo 5 > /sys/block/$variable1/queue/iosched/writes_starved
echo "2" > /sys/block/$variable1/queue/rq_affinity
echo 0 > /sys/block/$variable1/queue/rotational
echo 0 > /sys/block/$variable1/queue/add_random
echo 1024 > /sys/block/$variable1/queue/max_sectors_kb
echo 0 > /sys/block/$variable1/queue/nomerges
blockdev --setra 0 /dev/$variable1
```

Tuning Oracle

The next logical step in tuning an Oracle database server is applying tuning settings to the Oracle database itself. There are many possible database variables to set to configure an Oracle database. For an OLTP database, for example, the tuning options can be quite different than the settings for a data warehouse/analytics type of database. In performing OLTP types of benchmarks, there are numerous books and articles on the Web that describe Oracle database parameter settings for your particular environment. The following settings that can be set to increase performance in either environment are:

```
filesystemio_options=SETALL
disk_async_io=TRUE
4GB online redo logs (minimum, could be set much larger depending on environment).
```

When enabling the Oracle Database Smart Flash Cache feature, the following `init.ora/spfile` parameters need to be added:

Using filesystems for Database Smart Flash Cache, from SqlPlus:

```
SQL> alter system set db_flash_cache_file='/mount_point/flash.dbf' scope=spfile;
SQL> alter system set db_flash_cache_size=250g scope=spfile;
```

Verify Smart Flash Cache settings are implemented:

```
SQL> show parameter flash
NAME                      TYPE                      VALUE
db_flash_cache_file       string                    /mount_point/flash.dbf
db_flash_cache_size       big integer               250G
```

Using ASM for Database Smart Flash Cache, from SqlPlus:

```
SQL> alter system set db_flash_cache_file='+disk_group/flash.dbf' scope=spfile;
SQL> alter system set db_flash_cache_size=250g scope=spfile;
```

Verify Smart Flash Cache settings set in the SPFILE:

```
SQL> show parameter flash
NAME                      TYPE                      VALUE
db_flash_cache_file       string                    +disk_group/flash.dbf
db_flash_cache_size       big integer               250G
```

Conclusion

Implementing the Nytro flash accelerator card into an Oracle database infrastructure can dramatically increase database performance. By following these simple tuning tips outlined in this paper, a successful implementation of the Nytro flash accelerator card with optimal performance can occur in most environments.

When implementing the Nytro products in the Oracle database infrastructure, it is possible to reduce the memory footprint of the Oracle database buffer because it can be less expensive to perform physical I/O against the Nytro card compared to using hard drives. This memory can then be used for other areas inside the Oracle database or to allocate to other databases on the server.

seagate.com

AMERICAS Seagate Technology LLC 10200 South De Anza Boulevard, Cupertino, California 95014, United States, 408-658-1000
ASIA/PACIFIC Seagate Singapore International Headquarters Pte. Ltd. 7000 Ang Mo Kio Avenue 5, Singapore 569877, 65-6485-3888
EUROPE, MIDDLE EAST AND AFRICA Seagate Technology SAS 16–18, rue du Dôme, 92100 Boulogne-Billancourt, France, 33 1-4186 10 00