Для начала создадим БД и объекты, необходимые для сбора метрик производительности SQL-сервера.

Для простоты, я не стал в скрипте указывать опции создания БД:

```
create database monitor -- Создаем БД
GO
use monitor
GO
create table perf_counters -- Создаем таблицу, куда будем записывать данные по
счетчикам
(
        collect_time datetime,
        counter_name nvarchar(128),
        value bigint
)
GO
CREATE CLUSTERED INDEX cidx_collect_time -- Индекс, чтобы потом было быстрее делать
select
ON perf_counters
(
        collect_time
)
GO
```

Значения счетчиков производительности будем забирать из системного представления sys.dm_os_performance_counters. В скрипте описаны самые популярные и жизненно необходимые счетчики, естественно, список можно расширить. Хотелось бы пояснить по поводу CASE'ов. Счетчики, которые измеряются в «что-то»/секунду — инкрементальные. Т.е. SQL сервер каждую секунду прибавляет текущее значение счетчика к уже имеющемуся. Чтобы получить среднее текущее значение нужно значение в представлении делить на аптайм сервера в секундах. Узнать аптайм можно запросом:

```
select DATEDIFF(SS, (select create_date from sys.databases where name = 'tempdb'),
getdate())
```

Т.е. найти разницу между текущим моментом и временем создания tempdb, которая, как известно, создается в момент старта сервера.

Метрику Granted Workspace Memory (KB) сразу перевожу в мегабайты.

Процесс сбора оформим в виде процедуры:

```
CREATE procedure sp_insert_perf_counters
AS
      insert into perf_counters
      selectgetdate() as Collect_time,
                  Counter = CASE    WHEN counter_name = 'Granted Workspace Memory
(KB)' then 'Granted Workspace Memory (MB)'
                                    ELSE rtrim(counter_name) END,
                  Value =     CASE  WHEN counter_name like '%/sec%'
                                          then
cntr_value/DATEDIFF(SS, (select create_date from sys.databases where name =
'tempdb'), getdate())
                                    WHEN counter_name like 'Granted
Workspace Memory (KB)%' then cntr_value/1024
                                    ELSE cntr_value
```

```
                                        END
      from sys.dm_os_performance_counters where
      counter_name = N'Checkpoint Pages/sec' or
      counter_name = N'Processes Blocked' or
      (counter_name = N'Lock Waits/sec' and instance_name = '_Total') or
      counter_name = N'User Connections' or
      counter_name = N'SQL Re-Compilations/sec' or
      counter_name = N'SQL Compilations/sec' or
      counter_name = 'Batch Requests/sec' or
      (counter_name = 'Page life expectancy' and object_name like '%Buffer
Manager%') or
      counter_name = 'Granted Workspace Memory (KB)'
GO
```

Далее создадим процедуру, которая будет выбирать данные из нашей логовой таблицы. Параметры end и start задают временной интервал, за который мы хотим увидеть значения. Если параметры не заданы, выводить информацию за 3 последних часа.

```
create procedure sp_select_perf_counters
      @start datetime = NULL,
      @end datetime = NULL
as
      if @start is NULL set @start = dateadd(HH, -3, getdate())
      if @end is NULL set @end = getdate()
      select
            collect_time,
            counter_name,
            value
      from monitor..perf_counters
      where collect_time >= @start
      and collect_time <= @end
go
```

Завернем sp_insert_perf_counters в задание SQL-агента. С частотой запуска — раз в минуту.
Скрипт создания джоба я пропущу, чтобы не захламлять текст. В конце выложу все в виде одного скрипта.

Забегая вперед, скажу что дело было в том числе и из-за банальной нехватки оперативной памяти, поэтому сразу приведу скрипт, позволяющий посмотреть «борьбу» БД за буфферный пул. Создадим табличку, куда будем складывать данные:

```
CREATE TABLE BufferPoolLog(
        [collection_time] [datetime],
        [db_name] [nvarchar](128),
        [Size] [numeric](18, 6),
        [dirty_pages_size] [numeric](18, 6)
)
```

Создадим процедуру, которая будет выводить использование буфферного пула каждой отдельной базой данных:

```
CREATE procedure sp_insert_buffer_pool_log
```

```
AS
      insert into Monitor.dbo.BufferPoolLog
      SELECT
            getdate() as collection_time,
            CASE WHEN database_id = 32767 THEN 'ResourceDB' ELSE
DB_NAME(database_id) END as [db_name],
            (COUNT(*) * 8.0) / 1024 as Size,
            Sum(CASE WHEN (is_modified = 1) THEN 1
                  ELSE 0 END) * 8 / 1024 AS dirty_pages_size
      FROM
        sys.dm_os_buffer_descriptors
      GROUP BY
        database_id
```

Грязные страницы = измененные страницы. Эту процедуру заворачиваем в джоб. Я поставил выполняться раз в три минуты. И создадим процедуру для селекта:

```
CREATE procedure sp_select_buffer_pool_log
        @start datetime = NULL,
        @end datetime = NULL
AS
        if @start is NULL set @start = dateadd(HH, -3, getdate())
        if @end is NULL set @end = getdate()
        SELECT  collection_time AS 'collection_time',
                        db_name,
                        Size AS 'size'
        FROM BufferPoolLog
        WHERE (collection_time>= @start And collection_time<= @end)
        ORDER BY collection_time, db_name
```

Отлично, данные собираются, историческая база копится, осталось придумать удобный способ просмотра. И тут нам на помощь приходит старый добрый Excel.

Я приведу пример для счетчиков производительности, а для использования буфферного пула можно будет настроить по аналогии.

Открываем Excel, заходим в «Данные» — «Из других источников» — «Из Microsoft Query».

Создаем новый источник данных: драйвер — SQL Server или ODBC для SQL Server или SQL Server native Client, нажимаем «связь» и прописываем свой сервер, выбираем нашу БД в параметрах, в пункте 4 выбираем любую таблицу (она нам не понадобится).

Кликаем на наш созданный источник данных, нажимаем «Отмена» и на вопрос «Продолжить изменение запроса в Microsoft Query?» нажимаем «Да».

Закрываем диалог «Добавление таблицы». Далее идем в «Файл» → «Выполнить запрос к SQL». Пишем `exec sp_select_perf_counters`. Нажимаем ОК, идем в «Файл» — «вернуть данные в Microsoft Excel».

Выбираем, куда поместить результаты. Рекомендую оставить две строки сверху для параметров.
Идем в «Данные» — «Подключения», заходим в свойства нашего подключения.

Переходим на вкладку «Определение» и там, где текст команды пишем exec sp_select_perf_counters?,?..
Нажимаем ОК и Excel предлагает нам выбрать, из каких ячеек ему брать эти параметры. Указываем ему эти ячейки, ставим галки «использовать по умолчанию» и «автоматически обновлять при изменении ячейки». Лично я эти ячейки заполнил формулами:

Параметр1 =ТДАТА()-3/24 (текущие дата и время минус 3 часа)
Параметр2 =ТДАТА() (текущие дата и время)

Далее кликаем на нашей таблице и идем в «Вставка» — «Сводная таблица» — «Сводная диаграмма».
Настраиваем сводную таблицу:

Поля легенды — counter_name,
Поля осей — collect_time,
Значения — value.

Вуаля! Получаем графики метрик производительности. Рекомендую изменить тип диаграммы на «График». Осталась еще пара штришков. Переходим на страницу с нашими данными, опять заходим в свойства подключения и выставляем в «Обновлять каждые X мин» значение по желанию. Думаю, логично выставить частоту равную частоте выполнения задания на SQL сервере.

Теперь данные в таблице обновляются автоматически. Осталось заставить обновляться график. Переходим во вкладку «разработчик» — «Visual Basic».

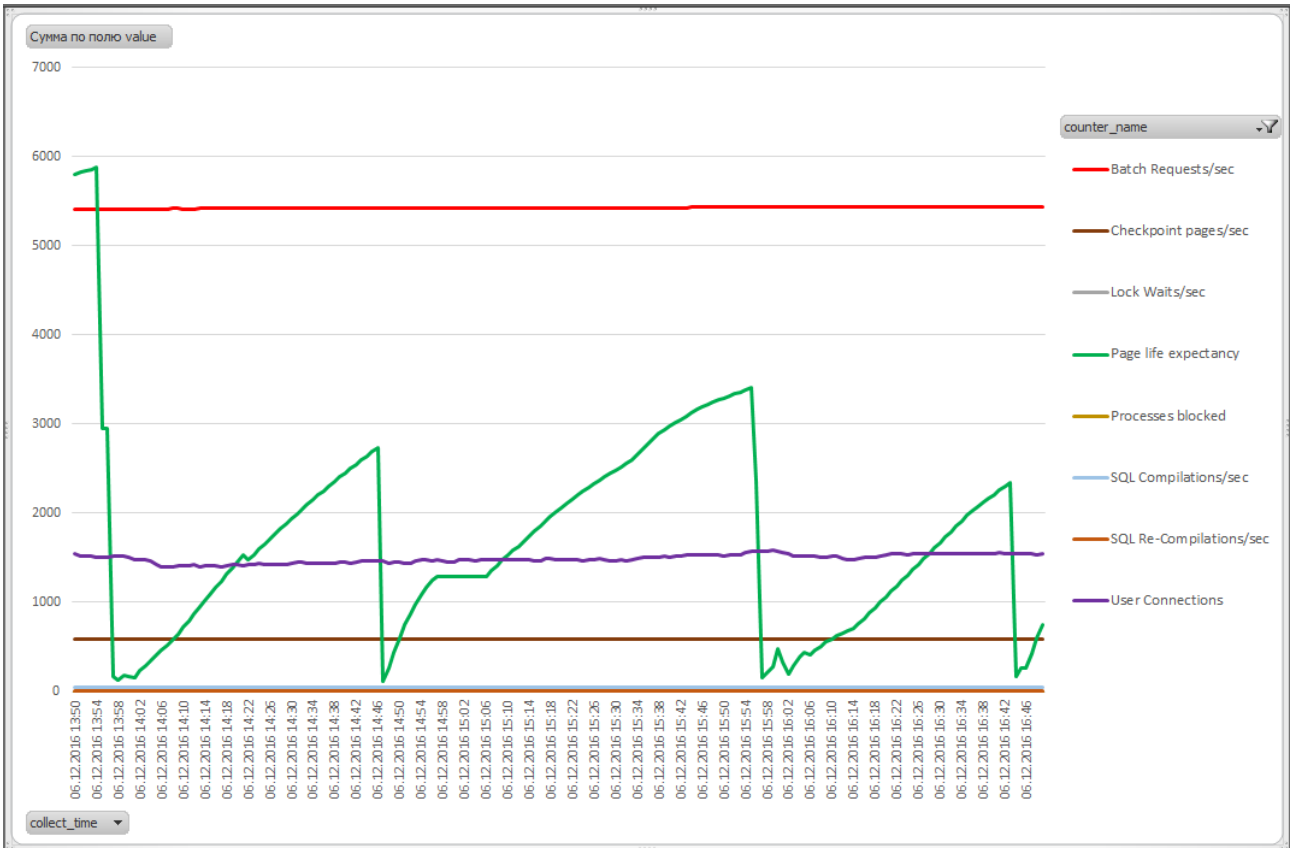
Кликаем слева на лист с исходными данными и вписываем следующий код:

```
Private Sub Worksheet_Change(ByVal Target As Range)
  Worksheets("Своднаятаблица").PivotTables("СводнаяТаблица1").PivotCache.Refresh
End Sub
```

де,

«Своднаятаблица» — имя листа со сводной таблицей. То имя, что указано в скобках в VB редакторе.
«СводнаяТаблица1» — имя сводной таблицы. Можно посмотреть, кликнув на сводной таблице и зайдя в раздел «Параметры».

Теперь наш график будет обновляться каждый раз, когда обновляется исходная таблица. Пример такого графика:



Для клонирования файла достаточно в свойствах нашего подключения в Excel изменить строку подключения, вписав новое имя сервера.

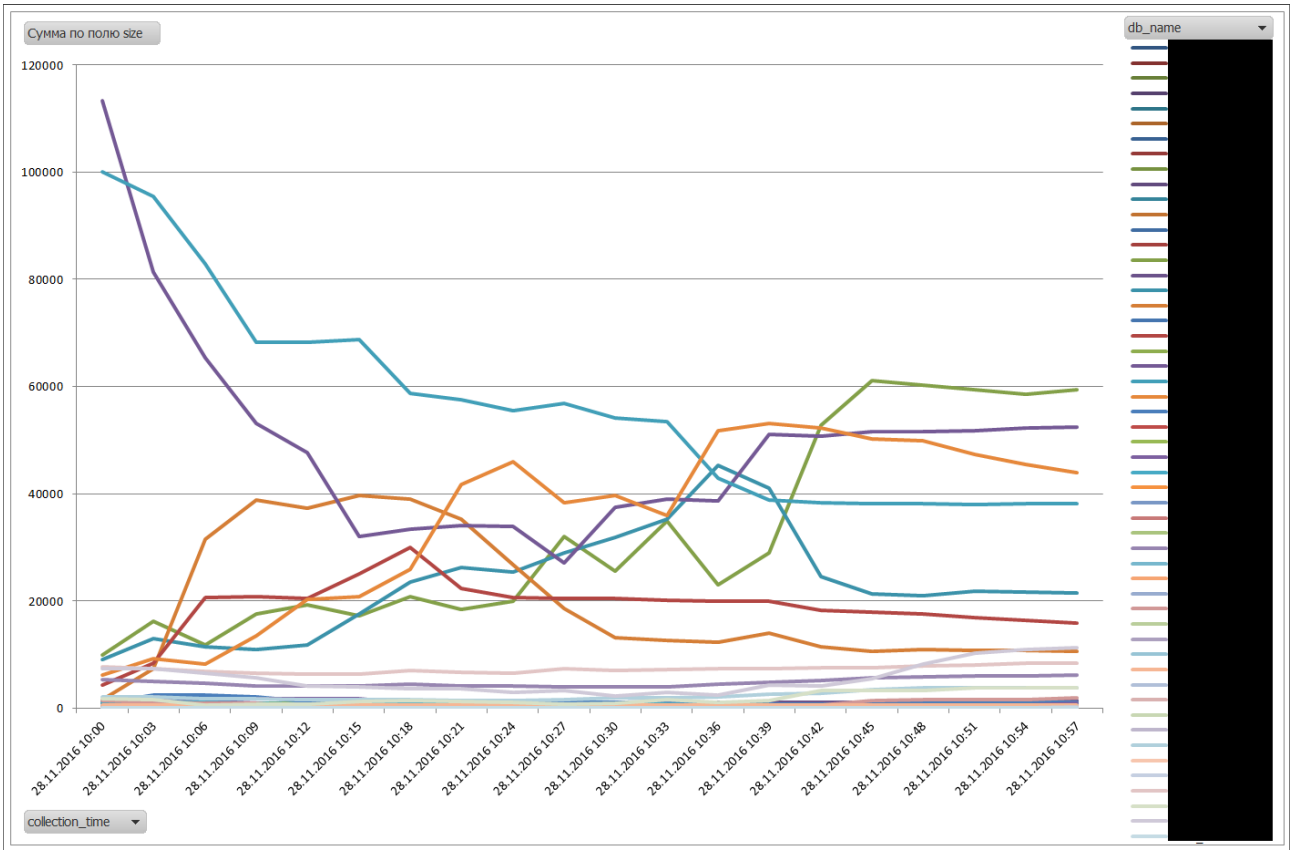
Касательно «борьбы» баз за буфферный пул и вычисления рекомендуемого количества оперативной памяти, для минимизации этой борьбы можно использовать следующий скрипт. Вычисляет максимальное использование оперативной памяти каждой БД, а также средний процент размера буфферного пула относительно общего размера оперативной памяти, выделенной серверу и на основании этих данных вычисляет «идеальный» размер оперативки, требуемой серверу:

```
DECLARE     @ram INT,
            @avg_perc DECIMAL,
            @recommended_ram decimal

--Узнаем, сколько сейчас выделено серверу
SELECT @ram = CONVERT(INT,value_in_use )
FROM sys.configurations
WHERE name = 'max server memory (MB)'
ORDER BY name OPTION (RECOMPILE);

--Узнаем какой процент от всей памяти составляет Buffer Pool
SELECT @avg_perc = avg(t.perc) FROM
(
SELECT sum(Size)/@ram*100 AS perc FROM Monitor.dbo.BufferPoolLog
GROUP BY collection_time
) t

--Вычисляем рекоммендуемый объем оперативной памяти
SELECT @recommended_ram = sum(t.maxsize)*100/@avg_perc  FROM
(
SELECT db_name, MAX(Size) AS maxsize FROM Monitor.dbo.BufferPoolLog
GROUP BY db_name
) t
```

```
select @ram as current_RAM_MB, @recommended_ram as Recommended_RAM_MB
```

Стоит заметить, что данные вычисления имеют смысл только если вы уверены, что запросы, работающие на сервере оптимизированы и не делают full table scan при каждом удобном (и не очень) случае. Также следует убедиться, мониторя метрику Maximum Granted Workspace, что у вас на сервере нет запросов, отъедающих часть буфферного пула под сортировку и hash-операции.

Пример борьбы баз за буфферный кэш (имена замазал):



Кстати, оказалось, что этот метод работает намного быстрее нашего заббикса