Power Dialing @ its Best

Your **WombatDialer** is quite a powerful tool: it is able to scale up significantly in order to manage **thousands of calls** at once. Whether you have 100 or 10,000 channels, you do not want to hit a wall the day you roll-out in production.

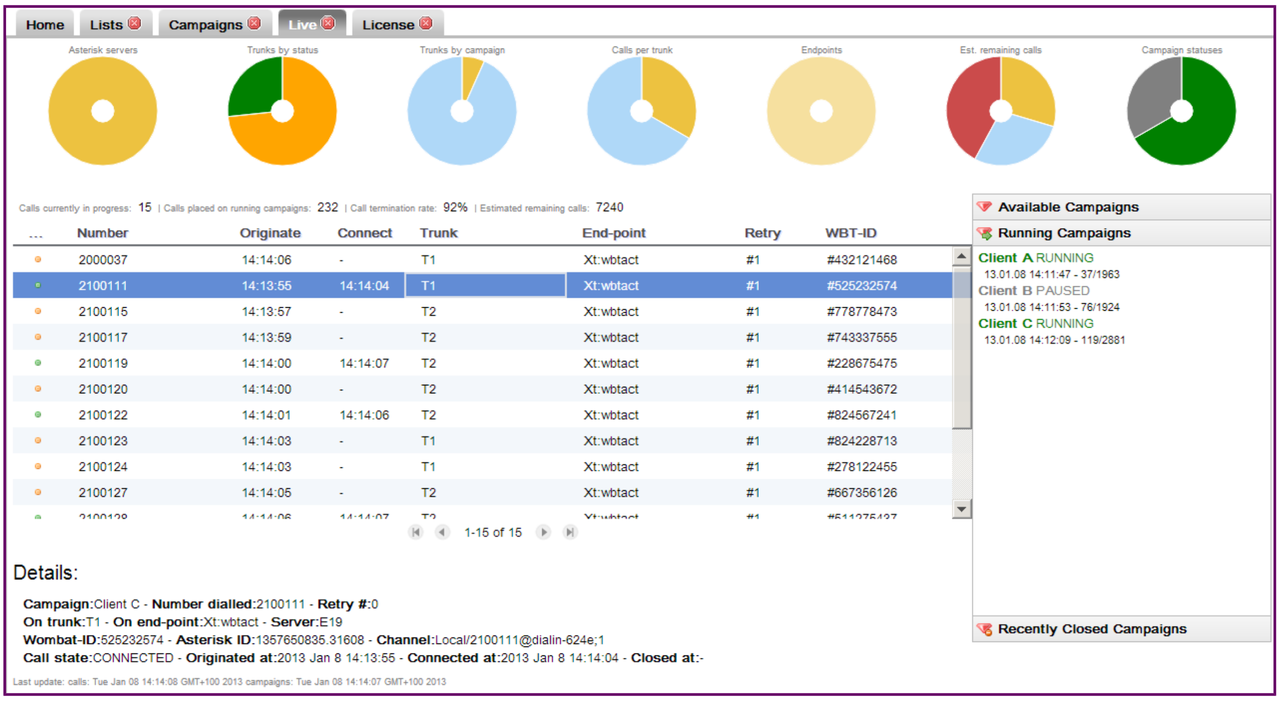
First, let's define what we mean by **"high-volume" dialing**.

For the purpouse of this article, we define "high-volume" as an amount of calls that will saturate a stock Asterisk system. It is hard to put a specific value here - it depends on the kind of hardware you use and what your PBX is actually doing when processing your calls. If you really want to put a number here, say it might be around 250 calls, plus or minus 150.

Optimizing Asterisk

As Wombat relies on Asterisk for dialing, it is important to understand what are the challenges to high-volume call processing on your Asterisk system:

First, what you do really matters. The number of calls that your PBX will handle successfully depends on what you are doing on it and how expensive each call is. There are three usual suspects here: transcoding, call recording and AGI scripts.



Calls currently in progress: 15 | Calls placed on running campaigns: 232 | Call termination rate: 92% | Estimated remaining calls: 7240

...	Number	Originate	Connect	Trunk	End-point	Retry	WBT-ID
●	2000037	14:14:06	-	T1	Xt:wbtact	#1	#432121468
●	2100111	14:13:55	14:14:04	T1	Xt:wbtact	#1	#525232574
●	2100115	14:13:57	-	T2	Xt:wbtact	#1	#778778473
●	2100117	14:13:59	-	T2	Xt:wbtact	#1	#743337555
●	2100119	14:14:00	14:14:07	T2	Xt:wbtact	#1	#228675475
●	2100120	14:14:00	-	T2	Xt:wbtact	#1	#414543672
●	2100122	14:14:01	14:14:06	T2	Xt:wbtact	#1	#824567241
●	2100123	14:14:03	-	T1	Xt:wbtact	#1	#824228713
●	2100124	14:14:03	-	T1	Xt:wbtact	#1	#278122455
●	2100127	14:14:05	-	T2	Xt:wbtact	#1	#667356126
●	2100128	14:14:06	14:14:07	T2	Xt:wbtact	#1	#511275437

Client A RUNNING
13.01.08 14:11:47 - 37/1963
Client B PAUSED
13.01.08 14:11:53 - 76/1924
Client C RUNNING
13.01.08 14:12:09 - 119/2881

Details:

Campaign: Client C - **Number dialled:**2100111 - **Retry #:**0
On trunk:T1 - **On end-point:**Xt:wbtact - **Server:**E19
Wombat-ID:525232574 - **Asterisk ID:**1357650835.31608 - **Channel:**Local/2100111@dialin-624e;1
Call state:CONNECTED - **Originated at:**2013 Jan 8 14:13:55 - **Connected at:**2013 Jan 8 14:14:04 - **Closed at:**-

Last update: calls: Tue Jan 08 14:14:08 GMT+100 2013 campaigns: Tue Jan 08 14:14:07 GMT+100 2013

Transcoding will use a lot of CPU to translate audio from some format to another. This happens silently every time you play a recording in a format and your channel uses a different format. Your music on hold is an MP3 and your channels speaks G729 - that's transcoding. While this is a very convenient feature Asterisk offers, it is a no-no for high volume dialing.

Call recording may get very expensive pretty soon, as it will use precious disk IOPS to write hundreds of audio files to disk and precious CPU cycles to convert audio into your favourite storage format. If you reall need call recording, you should make sure it is turned on only when needed (for example, a callee is in conversation with an agent), that your disks are as fast as you can get them (and a ramdisk is a good choice here) and that the storage format requires as little CPU as possible. For large scale recording, using a separate passive (network) recorder might be a good option.

AGI scripts require a process to be run on each invocation. They are often used in Asterisk as they provide a simple way to "glue" some logic and access to external services (webservices or databases) to the Asterisk dialplan. They are typically written in slow, interpreted scripting languages and will definitely not scale. When running Wombat, it is usually a better option to pre-fetch any value of interest coming from external systems ad having Wombat pass it to Asterisk as a channel variable. For example, instead of running an AGI script that launches an interpreter that opens a database connection over which you run an SQL query to decide whether you should play a message in English or in Spanish, this could be a channel variable set by Wombat that already contains the name of a custom audio file to play back.

The second big thing you should consider is **whether a GUI-based system is right for you**. Though having a GUI makes your life easier when administering the system, the kind of dial plan generated is often extremely byzantine and therefore slow. While this is done in order to offer you a ton of interesting features for your office PBX, you may find that you do not really need them all when you do large scale outbound dialing. The best of both worlds is using some hand-coded dialplan for dialing out (the part that is called by Wombat), and using your favourite GUI for all other functions.

A third area of concern is caused by the **AMI interface** that Wombat uses to drive Asterisk. The AMI protocol may get extremely chatty, as the system generates a lot of events, and this may lead to significant latency before all events are processed. This can be addressed by filtering the set of events that Asterisk sends (for example: Wombat does not need dialplan execution events), but even in this case at some point the problem will manifest itself. Plus, Asterisk can only process so many requests per second, and at some point you will be wanting to generate too many calls for your Asterisk system to handle.

One last and general issue with Asterisk is that **Asterisk processes voice channels**; if the system is overloaded and voice is not processed in real-time, you will encounter **voice quality issues**. Calling people and offering them a broken or unintellegible audio message is a big no-no. So you should make sure you have enough "capacity buffers" on your systems so that voice quality stays crystal-clear in every circumstance.

The real solution to high-volume dialing is therefore **scaling out**. This means running multiple Asterisk servers in parallel, each of which is processing only a limited set of channels that is well within its limits. Luckily, Wombat was designed from the start to **handle a pool of Asterisk servers as if they were one single, larger machine**. For Wombat there is almost no difference in processing 500 channels on one large Asterisk instance or 100 channels each on five different instances. This makes it possible to build real-world solutions that scale to the size of your problems.

Running large WombatDialer instances

Of course, Wombat has to work hard to drive a lot of calls at once. It has to retrieve numbers and attributes from multiple lists, clear them against blacklists, keep track of recalls, and track live calls and live queues as each of them sends their own events. Plus, Wombat syncs the current state to a database quite often, so that in case something goes severey wrong and it crashes, it can restart exactly from where it left without losing any calls. This is a lot of hard work, and happens in real-time.

When running Wombat, there are a few item that have to be tuned to make sure that everything runs smoothly; namely:

- System I/O and database load
- Dial latency and command queues
- Network latency
- JVM memory and GC policies
- Humans!

System load

A running WombatDialer system can only be as fast as the database it syncs to. Though Wombat uses batching to send the database an efficient set of updates, at some point you are going to notice that there is significant I/O wait on the system. You **do not want I/O wait to go over 5-10%**, because this makes Wombat less reactive; it takes more to fetch data and more to respond to changes on Asterisk, namely placing new calls. This can be mitigated by tuning the amount of memory that is available for MySQL to be used as a cache; the default settings that MySQL ships with are inadequate to run thousands of parallel calls. A **faster disk layer** (using SSDs and separate data disks) also helps a lot and can make all the difference.

Plus, Wombat has to fetch data from disk in order to know which calls are to be placed next. It does so by processing **batches of calls**, as this is way quicker than requesting each call separately, This means that when a channel is freed, Wombat already has a cached call ready to be placed. This value is controlled by the “*Batch size*” parameter on each campaign. A smaller batch size will cause Wombat to do more database work, but will cause less latency to be added. We suggest using a value that is about 2x the number of possible channels for a given campaign.

Dial latency

Dial latency is the amount it takes for Asterisk to “confirm” a call after it has been requested. On a smoothly running system, very few calls should be appearing on the Live page in state *REQUESTED*. Those are calls that have been requested to Asterisk but have not yet been acknowledged. Dial latency can be split into two parts:

- The time it takes for a requested call to reach Asterisk. Asterisk cannot process an infinite set of requestes per second, and will likely crash when overloaded. Wombat tries to mitigate this by dividing a command queue into “time units”, where only so-many items can be processed in each unit. For example, the default is to avoid sending more than 5 commands every 50 milliseconds. This is a conservative estimate, and depending on your Asterisk hardware a limit of 10 or event 20 messages every 50 milliseconds might be working for you. Having multiple Asterisk servers means that there are separate messages queues connecting to each PBX instance and so each queue only holds a fraction of all outgoing calls.

- The time it takes for Asterisk to confirm that a requested call is being processed. If this time starts being in the order of magnitude of seconds, the system is likely overloaded. It is okay for confirmation times to spike up when starting a campaign with a lot of channels to fill at once; it is not okay to have them consistently high all of the time.

The time it takes for a requested call to reach Asterisk. Asterisk cannot process an infinite set of requestes per second, and will likely crash when overloaded. Wombat tries to mitigate this by dividing a command queue into “*time units*”, where only so-many items can be processed in each unit. For example, the default is to avoid sending more than 5 commands every 50 milliseconds. This is a conservative estimate, and depending on your Asterisk hardware a limit of 10 or event 20 messages every 50 milliseconds might be working for you. Having multiple Asterisk servers means that there are separate messages queues connecting to each PBX instance and so each queue only holds a fraction of all outgoing calls.

The time it takes for Asterisk to confirm that a requested call is being processed. If this time starts being in the order of magnitude of seconds, the system is likely overloaded. It is okay for confirmation times to spike up when starting a campaign with a lot of channels to fill at once; it is not okay to have them consistently high all of the time.

You can keep track of the combined time by looking at the value named "*Wait Pre*" in the reports, and by observing the percentage of calls in state *REQUESTED* on the Live page. Watch out for maximum and average "Wait pre" times on your campaigns.

Additional **network latency** may be added by the network layer; Wombat is designed to work best with a local database and local Asterisk servers. Turn-around time to the database and to the Asterisk servers should be ideally zero milliseconds; values higher than 100 milliseconds (e.g. running WombatDialer in the US with Asterisk servers in Europe and a database in Singapore) will not work for high-load scenarios. Also, the bandwidth between all machines should be data-center or LAN class; consider that each large Asterisk instance might generate *4-5 megabit* of events per second.

Memory and GC

As per *memory*, WombatDialer does not really need a lot of memory to run into - you can have the engine process thousands of parallel calls in as little as 256M of heap. This said, if you run reports, have multiple users accessing the system, keep the Live page open and upload new call lists at the same time, the amount or RAM needed might be significantly higher. Plus, as with all Java applications, having less available memory means that the JVM has to be more aggressive in garbage collections. This leads to very uneven latency and it may even cause "stop the world" garbage collection events. This is something you want to avoid at all costs; memory is very cheap, so you should give Wombat a good amount or RAM to run into, and use a *throughput-oriented generational garbage collector* that is optimized to avoid major collection events at the price of using more working memory.

Usage patterns

The last item on our list are *human usage patterns*; the less activity you will have on a system at peak dialing time, the better. Don't leave the Live page open "just in case". Try to avoid running major reports or uploading very large lists when your system is fully loaded - most likely you can do it at a later time. Wombat includes a very extensive API that allows for complete remote control of Wombat instances.

Wrapping up

This said, it is reasonable to **run thousands of parallel calls on a WombatDialer 0.9 system with modern, high-end hardware**. When releasing Wombat 0.9, we did some significant tuning and optimization that means WombatDialer is about *2x as efficient* as previous versions were.

In any case, remember to do a **significant load test before going into production**. Whether you have 100 or 10,000 chnnels, you do not want to hit a wall the day you roll-out in production.

Dialing thousands of channels may not be your piece of cake, but it's nice to know that you can do it at the touch of a button. We believe that a good tool should give you margins for growth and preserve your existing investment and expertise. We work hard to make sure you will not have to worry about it.

POWER TO THE WOMBAT!