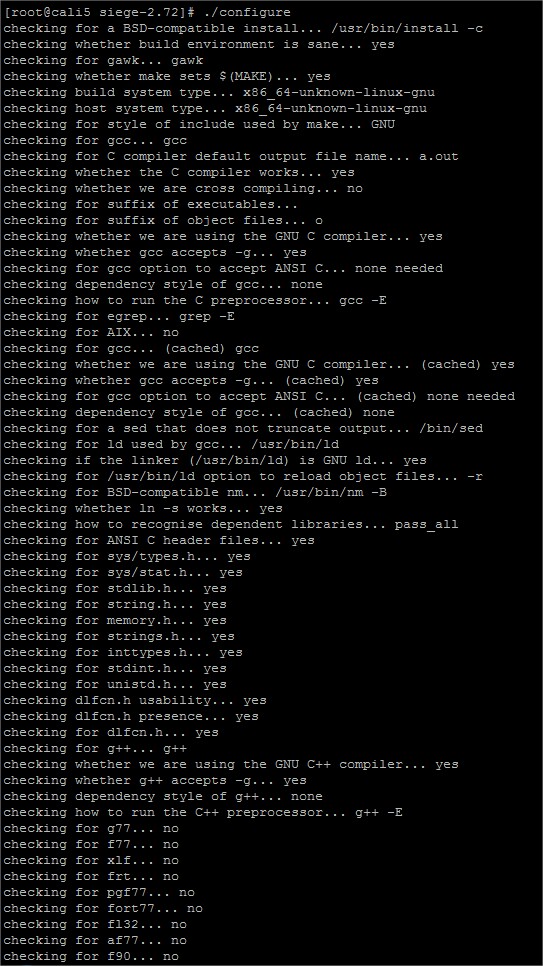
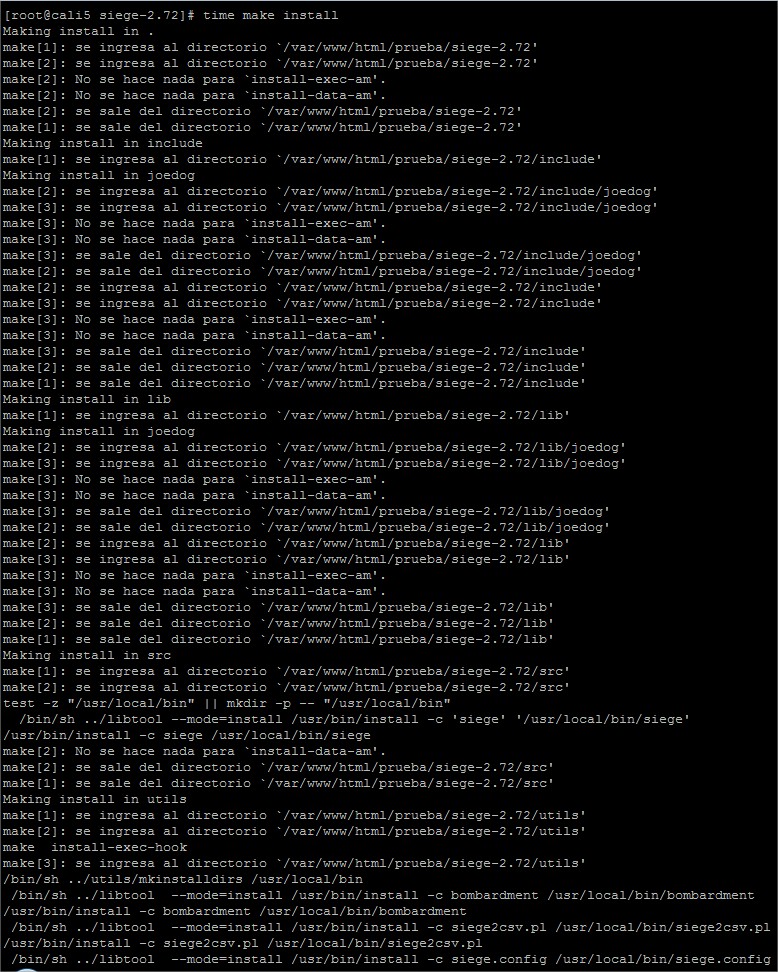
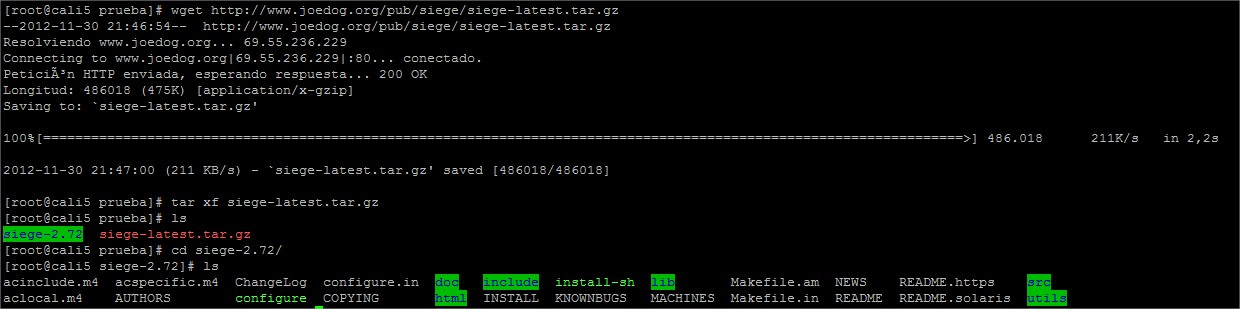
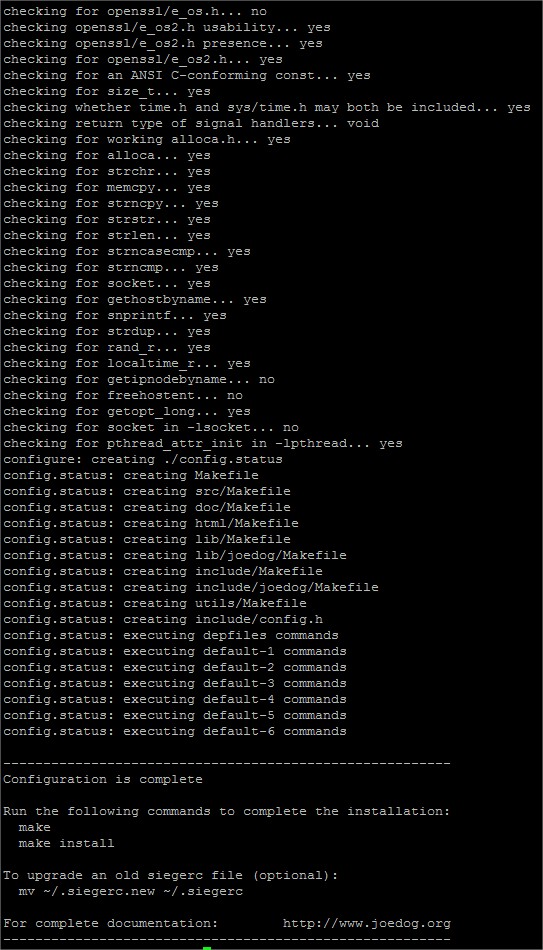
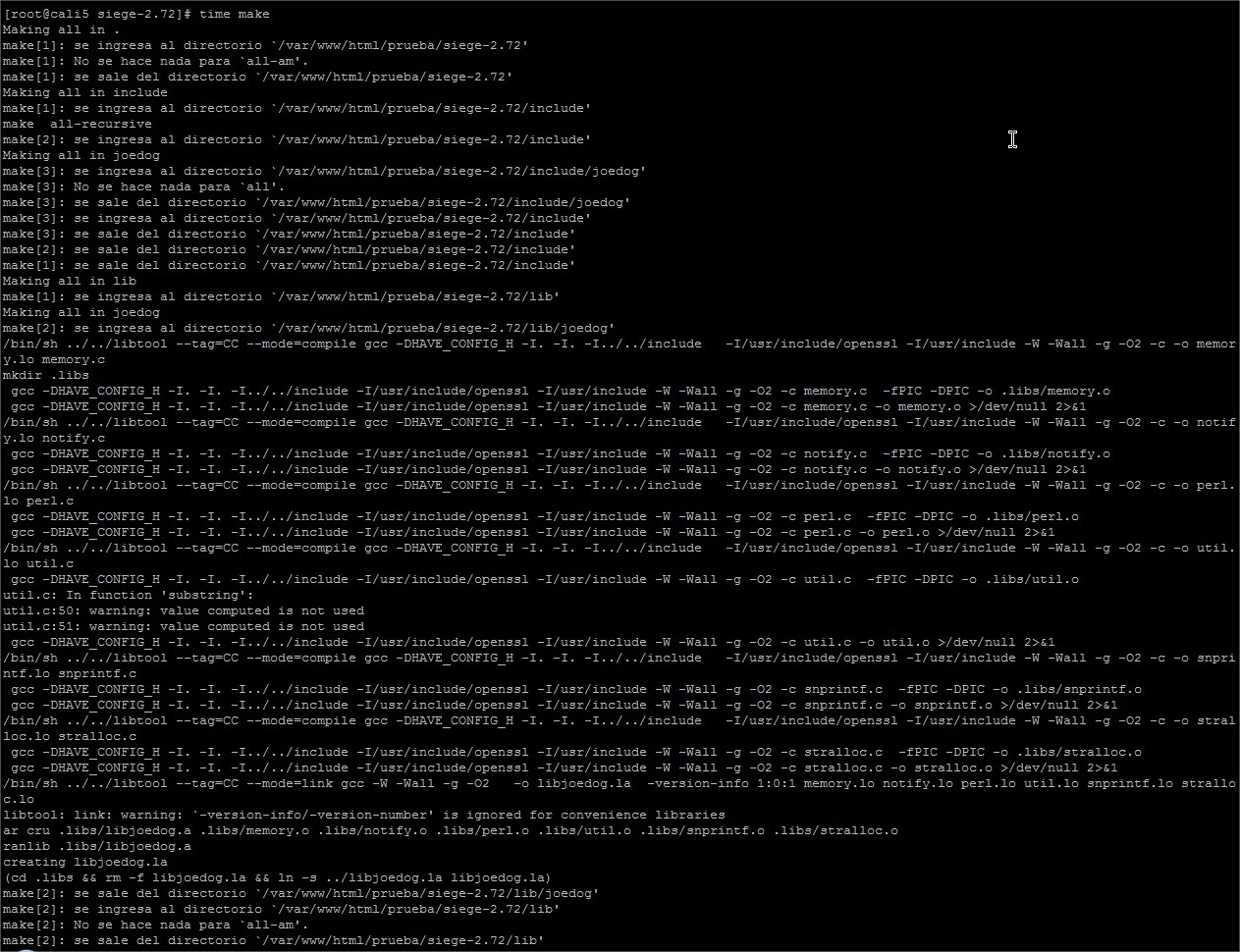
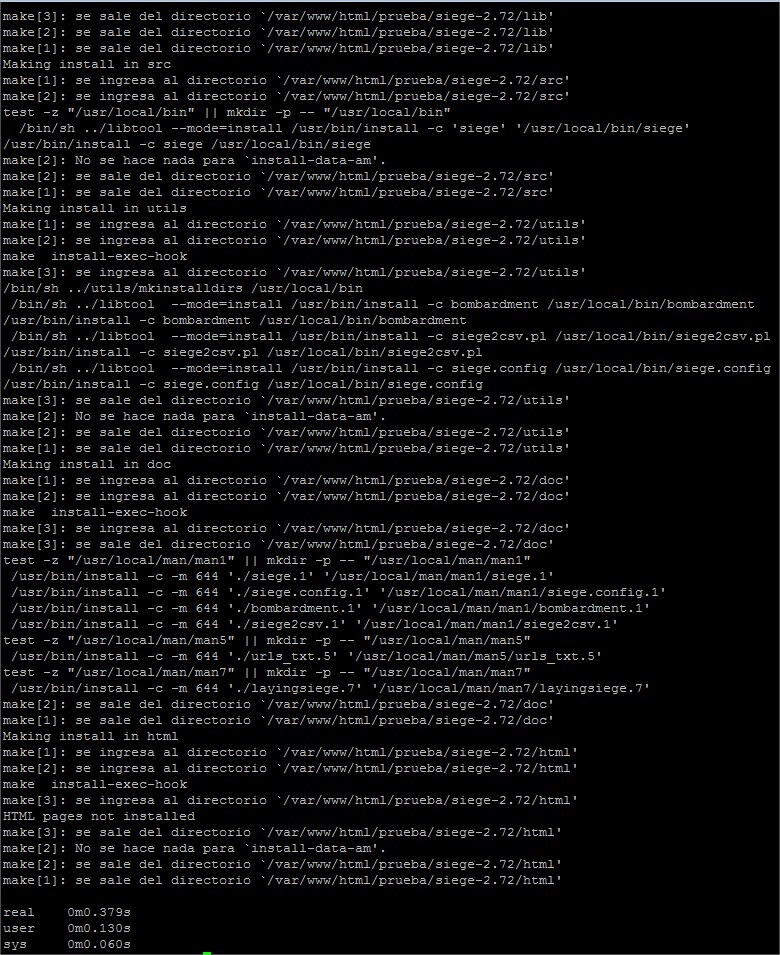
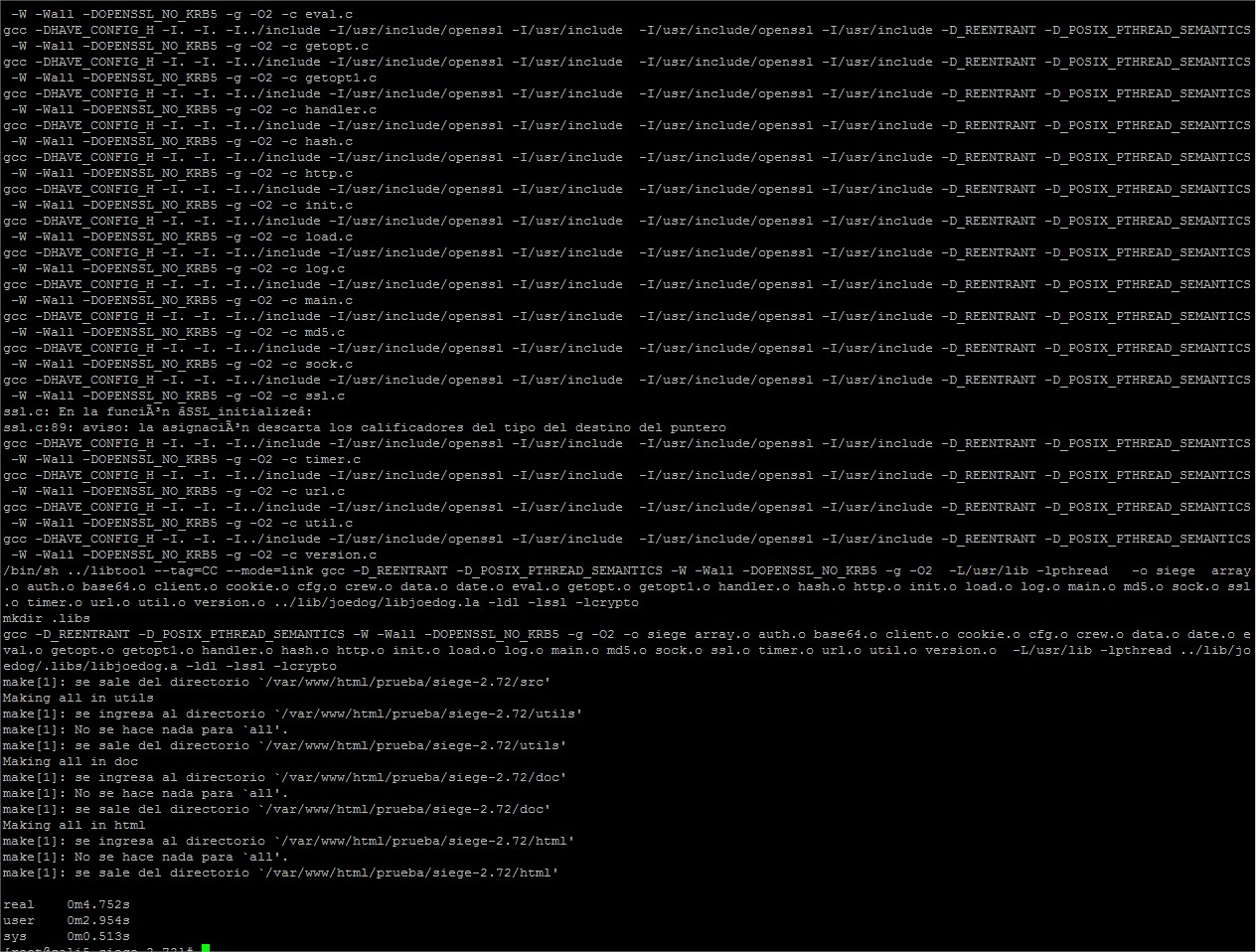
|  |
| --- |
| E:\Gustavo Adolfo\Dropbox\UNIVALLE\UNIVALLE.jpg |
|  |
|  |
|  |
|  |
|  |
| **James Andrés Payan Caicedo**  **cod: 0957150 plan: 2711**  **e-mail: jamespayan@gmail.com**  **José Antonio Nobile Rendón**  **cod: 0747102 plan: 2711**  **e-mail: jose.nobile@gmail.com**  **María Cristina Portilla Cortes**  **cod: 0844113 plan: 3743**  **e-mail:** [**macripco1311@gmail.com**](mailto:macripco1311@gmail.com) |
|  |

|  |
| --- |
| **FUNDAMENTOS DE REDES**  **Escuela de Ingeniería de Sistemas y Computación**  **Santiago de Cali**  **NOVIEMBRE de 2012** |

1. INSTALANDO SIAGE

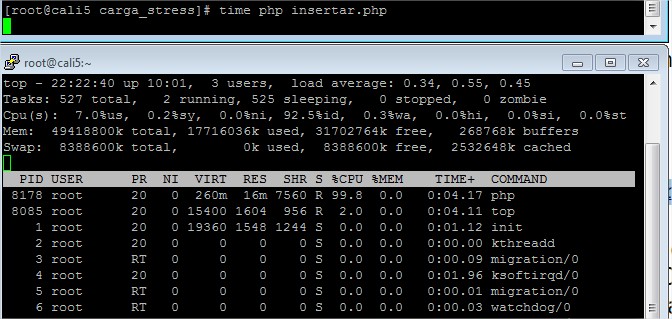


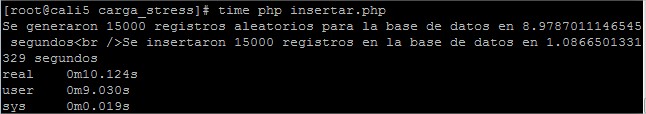




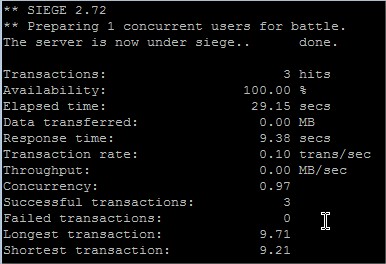
2)PRUEBAS

Aquí vemos que la generación de los 15mil registros aleatorio tarda 8.979 segundos, e insertarlos en un insert multiple 1.087 segundos, el tiempo total es 10.124 segundos, cuando se ejecuta, vemos que el proceso php usa el 100% de la CPU( esto corresponde a un núcleo) y usa 16MB de memoria reservada.

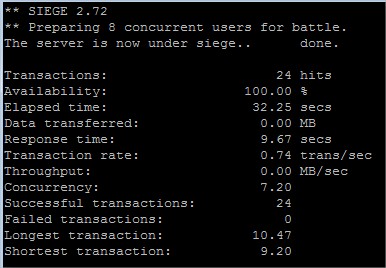




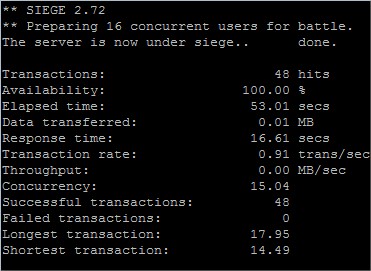
Realizo la prueba con 3 peticiones con intervalo de 1 segundo, simulando 1 usuario, solo para determinar cuando tarda ejecutándose, aquí la diferencia con la anterior prueba aparte del número de peticiones, es que la anterior es ejecutando directamente el php, y aquí es a través de Apache conectado a PHP usando fast CGI con PHP-FPM que es la última tecnología que han producido los desarrolladores de PHP para acoplar de la mejor forma a PHP con Apache, poniéndolo en un espacio de memoria reservado, que es seguro en entornos multithread, pudiendo estar apache configurado para usar threads (MPM – Worker) sin que esto implique un riesgo de problemas de memoria en PHP.



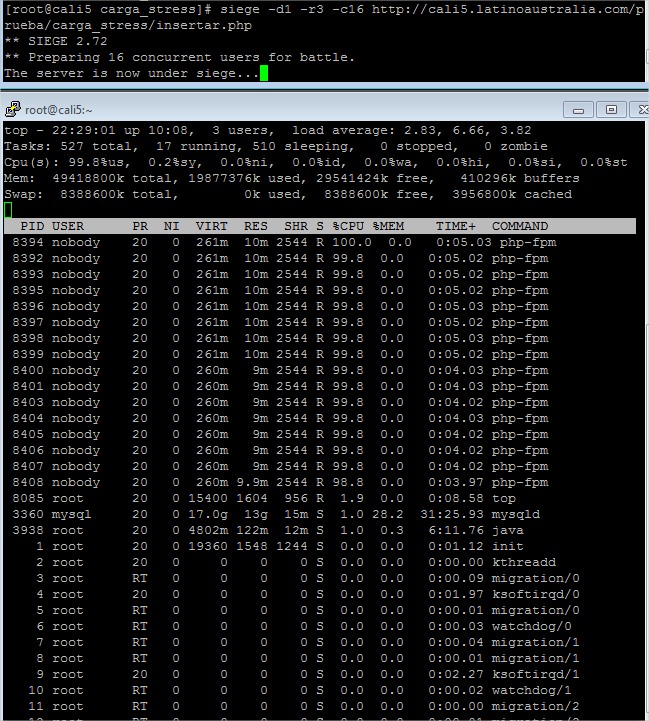
Aquí realizamos la prueba con 3 peticiones por cada uno de los 8 usuarios concurrente simulados, se observa que el tiempo de respuesta permanece igual, es decir el servidor responde igual con 8 usuarios concurrentes que con 1.



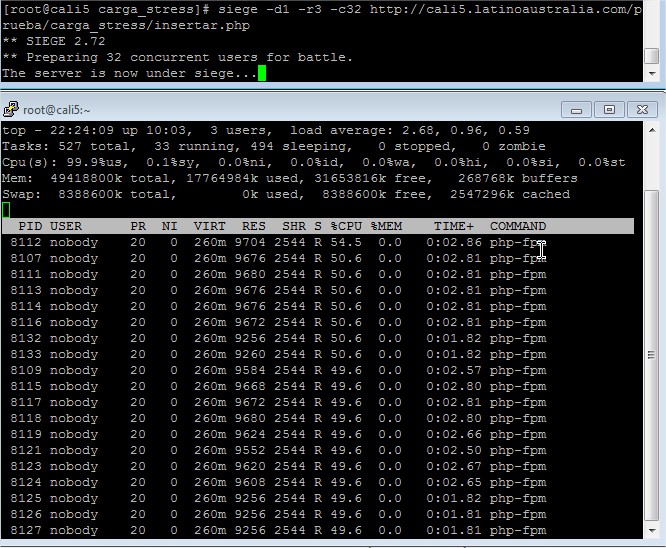
Aquí realizamos la prueba con 16 usuarios, vemos que la velocidad de respueta ha subido de promedio 9.7 segundos a 16.6 segundos, aquí ya vemos que empieza a afectarse negavitamente el rendimiento de la aplicación en el servidor.



Aquí una captura que muestra el proceso de PHP (PHP-FPM) ejecutándose con las 16 peticiones concurrentes, en este caso vemos que el porcentaje de tiempo inactivo es 0%, lo que significa que se ocuparon todos los núcleos del servidor.



En esta prueba con 32 usuarios, ya el servidor crashea, el apache empieza a no responder las peticiones, la prueba no finalizó correctamente.



Con esto puedo concluir, que a razón que el servidor tiene 2 procesadores con 4 cores y cada uno emula 2, para un total de 16 cores, el limite se alcanza cuando se ocupan los 16 cores, ya que el script que se está ejecutando demanda toda la velocidad de procesamiento de al menos un nucleo, aquí lo más demorado es generar los registros aleatorios, no tanto la inserción de los registros.

MODELOS DE DATOS

