Performance evaluation

Sequential:

Per al servidor seqüencial hem començat amb 5 clients incrementant de cinc en cinc, després de 25 a 50 i duplicant.
Com que el limit_listen el tenim a 10, al tenir 15 clients que volen escriure, s'han d'esperar a la cua y augmenta considerablement la latencia de les peticions.
Conforme més clients intenten connectar alhora al servidor, més brusc és l'augment

Unbounded:

Com al seqüencial, anem incrementant de cinc en cinc al principi, per després pujar a 25 i finalment anem duplicant els clients.
Els clients s'executen en paral·lel i sense límit, per això el temps mitjà que triga cada client no varia gaire entre 5 i 1000 clients: passem de 11 a 19 msec.
Com es pot veure a la gràfica de "Sockets results", el punt de saturació està en el els 200 clients.

Bounded:

Fem la mateixa estratègia que en els dos casos anteriors, amb la particularitat que hem limitat a 25 el número màxim de fills que es crearán.
En aquest cas sembla que a 200 clients s'estabilitza, però segueix creixent fins als 400, segurament perquè en aquest punt moren molts fills i el sistema perd temps gestionant-ho. És en aquell punt que comença a disminuir fins a estabilitzar-se en els 18 segons, igual que el unbounded.

Threads:

No hem pogut provar els threads perquè no ens executava el codi bé del tot. Hem estat debuggant i hem arribat a la conclusió que el problema està en que els threads i els clients no estan ben sincronitzats, doncs el client es queda penjat fent un read al socket de connexió.
Pensem que pot ser perquè el thread s'executa molt més ràpid que els clients (tenint en compte que els clients es creen amb un execlp) i que per tant quan fan el read on haurien de rebre "hola" simplement es desbloqueen perque el procés que hi ha d'escriure encara no ha sigut creat del tot.