	물리 서버의 상태에 따라 설정 값들을 최적화 시키기			
명칭	동일의미	동일의미	의미	사용처
Spark executor	YARN container		여러개의 task를 동시에(concurrent) 부릴 수 있다	
			spark에서 어떤 데이터를 broadcast 했을 때 동일한 JVM (= 동일한 executor)의 task들이 해당 데이터를 공유하게 된다	
			--num-executors == spark.executor.instances	
			--num-executors 인자값	spark-shell, spark-submit
			spark.executor.instances 설정값	spark-defaults.conf
task	core	vcore	각 executor가 사용하는 thread의 수	
			경험적으로 한 executor당 5 보다 작게 설정 할 때, 가장 성능이 좋다 (1보다크고)	
			너무 많으면 context switching 등이나 HDFS I/O 때문에 성능이 떨어짐	
			너무 적으면 하나의 JVM을 공유하는 장점이 사라짐	
			--executor-cores = spark.executor.cores	
			--executor-cores = spark.executor.cores	spark-shell, spark-submit
			spark.executor.cores	spark-defaults.conf
			--executor-cores 1로 설정해서 한 executor가 하나의 task만 실행하게 하면, 장점을 잃는다	
			더 수가 많고 작은 executor 들이 똑같은 데이터의 사본을 갖게 되는	
우리가 가진 서버(Nodes) 와 CPU 개수에 맞추어 설정 CPU 설정				
(executor 수) * (task 수) < (node 수) * (각 node의 CPU 개수)				
memory			--executor-memory == spark.executor.memory	
			YARN이 어떻게 설정했느냐에 따라 제약을 당할 수 밖에 없다	
			각 서버(node)가 가진 물리 메모리 수를 해당 서버가 책임질 것으로 예상되는 executor의 수로 나눈 값으로 설정	
CPU와 마찬가지로, 시스템을 위한 물리 메모리 1GB 이상을 늘 남김				

시스템구성				
총 서버는 총 6대, 각 서버가 가진 CPU는 16개, 각 서버가 가진 메모리는 64GB				
나쁜예	--num-executors 6 --executor-cores 15 --executor-memory 63G			
좋은예	--num-executors 18 --executor-cores 5 --executor-memory 19G			
> 노드마다 대략 3개의 executor가 수행				
> 각 executor는 5개의 core(=task)를 갖게됨				
> 전체 시스템 CPU개수인 6 * 15 = 90 보다 작다				
> --executor-memory 는 63.0 / 3 - 2 = 19 로 계산				
> 설정하지 않으면 Spark는 default로 1 core, 1GB 로 설정				