**Clustering Options**: I think the conceptual buzz word that covers this type of operation is "clustering". Averaging is by far the simplest to implement and works well for most purposes. The only time I would use something else is if you are worried about outliers [Edit]-> or the poles or the international dateline. [Edit]--> also averaging, while it will give you something that looks close to the center of the cluster, will be a little bit off because of projection inaccuracies caused by the fact that degrees lat lng aren't always the same distance apart in km/miles. The larger the area you average over the more the distortion.

Here is a comparison of a few clustering options

**Average (easy, fastest, inaccurate):** just sum the lat values & divide by the count and do the same for the lng values. Be sure to look out for overflow if you are using an Int32 some systems (notably c#) will silently overflow back to the low numbers. You can avoid these erros by using floating point precision for your sum accumulator. One problem with this method is that outliers could skew your location. [Edit]-> Another is that math near the poles and international date line doesn't average well and will skew locations badly.

**Nearest Neighbor (a little harder, slower, not outlier biased)** Rather than averaging you could go with the actual lat lng location with the smallest average distance to all its neighbors. This is kind of like taking a "median". The down side is that this is computationally expensive because you compare every point to every other point and calculate the distance between them. For example, clustering 10,000 points would require 100 million distance calculations.. not that slow but it definitely doesn't scale well.

**Grid Cell (needs a little extra setup, much faster, not outlier biased)** This is similar to nearest neighbor but much faster. You could pick an arbitrary level of precision, say .01 deg lat lng (which is about 1km roughly at populated lattitudes) and group your points into .01 x .01 degree buckets. You could then pick the bucket with the most points in it and either take the average of those points or run a nearest neighbor analysis on just those points. I use this method a lot with really big datasets (hundreds of billions of records) and find it a nice balance between precision and speed.

**Convex Hull Centroid (hard, slower, neat results):** You could also draw a band around your points to define a shape that covers them all ([see wikipedia](http://en.wikipedia.org/wiki/Convex_hull)), and then calculate the center point of this shape. Typical centroid functions aren't center weighted so you would need to so some kind of inverse nearest neighbor analysis using sample points inside your shape until you found the one furthest from the edges. This method is really more interesting because of the convex hull itself rather than the actual center finding algorithm which is neither fast nor particularly precise.. but the hull shape may have other useful applications with your data.